Benutzerprofile für die Anfrageverarbeitung in verteilten Digitalen Bibliotheken

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Bethina Schmitt

aus Erlangen

Tag der mündlichen Prüfung:	9. Juli 2003
Erster Gutachter:	Prof. Dr. Peter C. Lockemann
Zweiter Gutachter:	Prof. Dr. Frieder Nake

Danksagung

Die vorliegende Dissertation entstand während meiner Tätigkeit am Institut für Programmstrukturen und Datenorganisation an der Universität Karlsruhe. Die Thematik ergab sich aus der Zielsetzung des DFG-Forschungsprojekts UniCats und aus meinem persönlichen Interesse an benutzerzentrierten Ansätzen. Zum Gelingen meiner Arbeit haben zahlreiche Personen direkt oder auch indirekt beigetragen, denen allen möchte ich an dieser Stelle danken.

Mein besonderer Dank gilt meinem Doktorvater, Herrn Professor Peter Lockemann. Er ließ mir bei meiner wissenschaftlichen Arbeit viele Freiräume für die Entwicklung eigener Ideen und Visionen und verfolgte meine Arbeit dabei stets mit großem Interesse. Durch zahlreiche Denkanstöße und neue Impulse in der Anfangsphase sowie wertvolle, stets konstruktive und motivierende Anmerkungen in den folgenden Phasen konnte die Arbeit schließlich diese Qualität erlangen. In der Endphase ermöglichte seine rasche und sorgfältige Durchsicht dann auch ein zügiges Fertigstellen der Arbeit. Ich bewundere sein Arbeitspensum und habe die Zusammenarbeit mit ihm sehr genossen. Er wird immer ein großes Vorbild für mich sein.

Weiterhin möchte ich Herrn Lockemann für die Ermunterung und die Unterstützung zahlreicher internationaler Veröffentlichungen und der damit verbundenen Konferenzbesuche danken, die mir die Möglichkeit gaben, das internationale Forschungsgeschehen aus der Nähe zu erleben und mitzugestalten. Viele interessante Begegnungen und Diskussionen haben mich in meiner wissenschaftlichen Arbeit inspiriert und mich auch persönlich bereichert und weitergebracht.

Mein herzlicher Dank geht an Herrn Professor Frieder Nake. Er war sofort zur Übernahme der Zweitbegutachtung bereit und hat meine Arbeit stets neugierig und wohlwollend begleitet. Durch seine erfahrenen Kommentare konnte die Arbeit im Bereich der Benutzungsunterstützung besser positioniert und der Kandidatin vor der mündlichen Prüfung zu deutlich mehr Gelassenheit verholfen werden.

Herr Professor Fellner ist Koordinator des DFG-Schwerpunktprogramms V^3D^2, an welchem das UniCats-Projekt beteiligt ist. Ihm möchte ich für das in uns gesetzte Vertrauen danken und dafür, dass er unsere Arbeit mit besonderem Interesse verfolgt und uns gerade auch bei unkonventionellen Ansätzen wie QuakeBib Mut gemacht und unterstützt hat.

Allen aktuellen und ehemaligen Mitarbeitern des Lehrstuhls danke ich für ihre stetige Diskussionsbereitschaft und das angenehme Arbeitsumfeld. Stellvertretend seien die Kollegen genannt, mit denen mich eine längere gemeinsame Zeit am IPD verbindet, Patricia Krakowski, Dr. Birgitta König-Ries, Matthias Gimbel, Christian Weinand, Jens Nimis, Dr. Khaled Nagi und Dr. Jörg Schlösser. In besonderer Weise möchte ich meine Kollegen aus dem UniCats-Projekt erwähnen, Dr. Sebastian Pulkowski und Michael Christoffel. Die

Zusammenführung unserer unterschiedlichen Stärken hat sich sicherlich nicht nur auf die Projektarbeit, sondern auch unsere jeweilige persönliche Entwicklung positiv ausgewirkt. Weiterhin geht mein herzlicher Dank an Dr. René Witte, der mir bei meiner ersten eigenen internationalen Veröffentlichung rat- und tatkräftig zur Seite stand und mir damit den Weg für alle weiteren Publikationen eröffnete.

Wesentlichen Anteil an dem angenehmen Arbeitsumfeld und damit letztlich am Gelingen der Arbeit haben natürlich auch alle Studenten, die im Rahmen des UniCats-Projekts und der Lehrveranstaltungen als Studienarbeiter, Diplomanden und wissenschaftliche Hilfskräfte mit mir zusammengearbeitet haben. Stellvertretend möchte ich an dieser Stelle Andreas Schmidt, Sven Oberländer und Jens Nimis nennen sowie das Tutorenteam des Datenbankpraktikums, insbesondere Frank Drewek, Jürgen Schneider und Rainer Vogt.

Und ich möchte mich natürlich bei meiner Familie und bei meinen Freunden bedanken. Meine Eltern haben mich dazu ermuntert, meine mathematischen und logischen Fähigkeiten im Rahmen eines Informatik-Studiums weiter zu entwickeln und stets an mich und schließlich an meinen Erfolg in der Wissenschaft geglaubt, dafür bin ich ihnen sehr dankbar. Meine Mutter war mir in all den Jahren ein liebevoller Ansprechpartner, meinem Vater danke ich, dass er mir trotz seiner fachlichen Kompetenz den Freiraum für meine persönliche Entwicklung ließ. Im beruflichen Umfeld habe ich gute Freunde gefunden, mit denen ich die Höhen und Tiefen einer Dissertation gemeinsam durchlebt habe, dafür danke ich Dr. Martin Hess, Dr. Holger Leuck und Wolfgang Hürst. Ich danke auch meinen Freunden, die nicht so viel von Informatik, aber umso mehr vom Genießen des Lebens verstehen, für die gemeinsamen Yoga- und Tanzabende und unsere geselligen Runden. Dadurch blieb mir auch in Zeiten der intensivsten Arbeit und Forschung der Bezug zum Leben erhalten.

Abschließend möchte ich mich besonders bei meinem Freund René Brunner bedanken, der in der gesamten Promotionszeit viel Rücksicht auf mich genommen hat, mir seelischen Beistand und tatkräftige Unterstützung gegeben hat und der mir ein wertvoller Gesprächspartner und geschätzter Berater geworden ist.

Karlsruhe, im Dezember 2003 Bethina Schmitt

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Suchmaschinen

Information und Wissen haben in den letzten Jahren einen bedeutenden Stellenwert im gesellschaftlichen Diskurs erhalten. In unserer Gesellschaft – bekanntlich auch gerne als Informationsgesellschaft bezeichnet – wird sich der Mensch mehr und mehr bewusst, dass er Informationen benötigt und dass ihm benötigte Informationen fehlen, sei es für das berufliche Weiterkommen oder auch für einfache Alltagsdinge. Die Suche nach und die Verarbeitung von Information sind im Begriff, sich zu neuen Kulturtechniken zu entwickeln.

Nicht ohne Grund hat das World-Wide Web (WWW) als Plattform für die Bereitstellung und den Abruf von Informationen unterschiedlichster Art einen so durchschlagenden Erfolg. Zum einen haben dadurch Gesellschaftsgruppen Zugang zu Informationen erhalten, die vormals allenfalls Fachleuten zugänglich waren, zum anderen ist die Informationssuche – gemessen am Ergebnis – mit beträchtlich weniger Mühen verbunden: Ein paar Mausklicks reichen aus, wofür vor nicht allzu langer Zeit ein schriftliches Auskunftsersuchen oder ein persönlicher Besuch notwendig waren.

Der Siegeszug des World-Wide Web hält nach wie vor an. Mittlerweile beeinflusst es zahlreiche Bereiche unseres Berufs- und Privatlebens und, obwohl es nun erst seit etwa zehn Jahren existiert, wäre es nicht mehr wegzudenken. Die Menge der verfügbaren Informationen nimmt auch weiterhin rasant zu, Berechnungen der Firma Cyveillance zufolge hat die Anzahl der verfügbaren HTML-Seiten im Juli 2000 die 2-Milliarden-Marke durchbrochen (das entspricht etwa 20 Terabyte an Daten) und an jedem Tag kommen weitere 7,3 Millionen Webseiten neu hinzu.

In Anbetracht dieser Informationsfülle kommt der Informationssuche jetzt und in Zukunft in besonderem Maße eine zentrale Bedeutung zu. Sie stellt einen kritischen Erfolgsfaktor dar. Nur wenn es gelingt, die Informationssuche so zu verbessern, dass die verfügbaren Informationen zum einen nutzbar und auffindbar sind und dass zum andern der Benutzer vor der Gefahr der Informationsüberflutung geschützt wird, wird die Gesellschaft von diesem reichhaltigen und vielfältigen Informationsangebot auch entsprechend profitieren können.

Zur Informationssuche stehen dem Benutzer mittlerweile eine ganze Reihe an Suchdiensten, sog. Suchmaschinen für das WWW, zur Verfügung. Zunächst sind einzelne Suchmaschinen wie Altavista, Northern Light und Google entstanden. Diese unterscheiden sich in der Menge der indizierten Seiten, der thematischen oder geographischen Ausrichtung, der Aktualität der indizierten Seiten und im verwendeten Retrieval Algorithmus. Doch trotz Einsatz eines immensen Rechner- und Rechenaufwands deckt jede dieser Suchmaschinen nur einen

begrenzten Ausschnitt des WWW ab. Daraus erwuchs nachfolgend die Technologie der Meta-Suchmaschinen: Meta-Suchmaschinen führen keinen eigenen Datenbestand. Sie setzen auf mehreren existierenden Suchdiensten auf, leiten eine Benutzeranfrage an alle eingebundenen Suchmaschinen weiter und geben dem Benutzer schließlich ein integriertes Gesamtergebnis zurück. Dadurch bieten Meta-Suchmaschinen dem Benutzer einen einheitlichen Zugangspunkt und gewährleisten eine deutlich größere Abdeckung des WWW als die Einzeldienste.

Mittlerweile haben sich viele Suchmaschinen spezialisiert. Beispielsweise gibt es Suchdienste für die gezielte Informationssuche nach Börsendaten, Nachrichten, Veranstaltungs- oder Reiseinformationen. Auch bereits bestehende Datenbank- oder Retrievalsysteme sind mit einer Schnittstelle zum Web ausgestattet worden, so dass ihre Funktionalität nun weltweit zur Verfügung steht. Derartige Systeme bezeichnet man auch oft als „Hidden Web“, da die nachgewiesenen Daten nicht auf statischen HTML-Seiten zu finden sind, sondern nur über das spezielle Suchsystem durch eine entsprechende Anfrage dynamisch abgerufen werden können.

1.2 Die wissenschaftliche Literaturrecherche

Eine spezielle Form der Informationssuche ist die wissenschaftliche Literaturrecherche: Dabei sucht ein Benutzer nach Büchern, Artikeln oder anderen Veröffentlichungen, die er zum Anfertigen seiner eigenen wissenschaftlichen Arbeit (z.B. einer Seminararbeit, Diplomarbeit oder Dissertation) benötigt. An eine wissenschaftliche Literaturrecherche werden höhere Ansprüche gestellt als an eine allgemeine Informationssuche, welche in der Regel weniger zielgerichtet verläuft und durchaus auch einen unterhaltenden Charakter haben kann. Die Unterschiede liegen zum einen in der Art der gesuchten Dokumente, zum anderen in der Durchführung der Suche selbst. Die gewünschten Dokumente sollen von gesicherter Qualität sein, d.h. sie sollen die für eine Veröffentlichung notwendigen Begutachtungsprozesse und Qualitätskontrollen durchlaufen haben. Die Suche selbst soll sorgfältig und systematisch durchgeführt werden, d.h. dabei sollen möglichst keine relevanten Informationen übersehen werden, insbesondere keine Standardwerke oder aktuellen Arbeiten. Meistens erstreckt sich dadurch eine wissenschaftliche Literaturrecherche zu einem Thema über mehrere Sitzungen und über einen längeren Zeitraum.

Durch den Siegeszug des World-Wide Web hat sich auch die wissenschaftliche Literaturrecherche erheblich verbessert: In den traditionellen Bibliotheken wurden die ehemaligen Zettelkataloge schon vor einigen Jahren von elektronischen Recherchesystemen, den sogenannten OPACs (Online Public Access Catalogs) abgelöst. Über eine Schnittstelle zum World-Wide Web stehen diese OPACs nun weltweit der Allgemeinheit zur Verfügung, unabhängig von den regulären Öffnungszeiten der Bibliotheken.

Auch das Dienstangebot, welches dem Benutzer neben den klassischen Bibliothekskatalogen zur Verfügung steht, ist reichhaltiger und vielfältiger geworden: Die meisten Verlage präsentieren ihre Dokumente mit diversen Zusatzinformationen, Online-Buchhändler bieten ein umfangreiches Sortiment an Büchern, häufig sogar in mehreren Sprachen, an, welche dann direkt über das Internet bestellt werden können. Mehrere Fachinformationszentren und Fachdatenbanken stellen ihre lückenlos gepflegten Bestände zur Verfügung, allerdings meist gegen eine Gebühr, und diverse Fachgesellschaften und universitäre Einrichtungen haben Volltextarchive mit Technischen Berichten und Forschungsartikeln zum sofortigen Herunterladen aufgebaut.

Auch wenn das Informationsangebot nun wesentlich attraktiver geworden ist, die Informationssuche selbst ist für einen Benutzer dabei nicht wirklich einfacher geworden. Früher war es eher problematisch, überhaupt etwas Geeignetes zu finden und sich dieses Dokument daraufhin zu beschaffen. Heute besteht das Problem vielmehr darin, aus der Menge der verfügbaren Dienste und Informationen den/die geeignetsten herauszufinden. Das ist häufig mit einer aufwendigen und langwierigen manuellen Vergleichsarbeit verbunden und setzt eine gute Kenntnis der verfügbaren Dienste und ihrer Leistungsmerkmale voraus.

Nachdem derzeit viele verschiedene Dienste zur wissenschaftlichen Literaturrecherche im WWW existieren, lässt sich derselbe Trend beobachten wie bei Suchmaschinen: Gefordert werden integrierte Systeme oder Meta-Recherchesysteme, um damit einen größeren Dokumentbestand einheitlich nutzbar zu machen. Dadurch muss der Benutzer die einzelnen Dienste zum einen nicht mehr kennen und zum anderen auch nicht mehr separat befragen.

Die aktuellen integrierten Recherchesysteme kann man in zwei Klassen einteilen: Die erste Klasse von Ansätzen beinhaltet zwar verschiedene Dokumentbestände, verwendet aber dieselbe einheitliche Technik dafür. Der Datenbestand der zugrundeliegenden Systeme wird entweder manuell oder maschinell in das Gesamtsystem im gewünschten Format eingespielt. Technisch gesehen entsteht genau genommen wieder ein monolithisches System mit eigenem Datenbestand. Für diese Vorgehensweise sind zum einen Absprachen mit den Einzeldiensten notwendig und es entsteht ein großer Integrations- und vor allem auch Pflegeaufwand im laufenden Betrieb für Aktualisierungen. Bekannte Beispiele für diese Vorgehensweise sind NCSTRL, DBLP, CSBIB oder der MeDoc-Ansatz.[1]

Die zweite Klasse von Ansätzen wahrt die Autonomie der eingebundenen Dienste und führt keinen eigenen Datenbestand. Dieser Ansatz entspricht technisch dem Vorgehen der Meta-Suchmaschinen im WWW. Dadurch ist das Meta-Recherchesystem in der Angebots-zusammenstellung sehr flexibel und kann seine Dienste beispielsweise umfassend oder für einen Bereich repräsentativ auswählen. Beispiele für diese Klasse sind KVK oder Daffodil.[1]

[1] Die vollständigen Namen und die Webadressen der in der Arbeit angesprochenen Literaturdienste sind in Anhang A zu finden.

Beobachtet man diese Ansätze allerdings genauer, fällt der geringe Integrationsgrad der einbezogenen Dienste auf. Lediglich die Informationen der Kurztitellisten werden im Gesamtergebnis präsentiert, Duplikate werden meist nicht ermittelt oder hervorgehoben. Die Vervollständigung der Informationen bleibt dem Benutzer selbst überlassen, welchen dabei nur zu oft das Lost-in-Hyperspace-Syndrom ereilt. Der Grund für den geringen Integrationsgrad liegt sicherlich in der Informationsportionierung der über das WWW eingebundenen Dienste. Eine weitergehende Integration ist auf Basis der klassischen Meta-Technologie nicht möglich, die Ergebniszusammenstellung würde extrem zeitaufwendig, da viele einzelne Web-Anfragen bearbeitet werden müssten. Für die Benutzer bedeutet dieser geringe Integrationsgrad eine erhebliche Belastung und die Synergien zwischen den Einzeldiensten werden auf diese Weise weder hervorgebracht noch gewinnbringend genutzt.

1.3 Ziele der Arbeit

Ziel der Arbeit ist die Entwicklung eines Meta-Recherchesystems für den Bereich der Literatursuche, welches die vorhandenen Informationen der Einzeldienste integriert. Der Benutzer soll mit möglichst vollständigen Informationen in seinem Recherche- und Entscheidungsprozeß unterstützt werden, ohne allzu lange auf die Ergebnisse warten zu müssen. In der Arbeit wird somit die Fragestellung untersucht, mit welchen Konzepten und technischen Lösungen eine gleichzeitige Steigerung von Integrationsgrad und Performanz (im Sinne von schnellen Antwortzeiten) erfolgen kann und wie weit diese reicht.

Dazu soll in einem ersten Schritt die Möglichkeit geschaffen werden, Suchanfragen so präzise zu formulieren, dass einer drohenden Informationsüberflutung des Benutzers entgegengewirkt wird. Die Präsentation der Ergebnisse soll den Entscheidungsprozess des Benutzers in geeigneter Weise unterstützen. Die Interaktion soll für den Benutzer einerseits möglichst einfach und intuitiv geschehen, andererseits soll sie auch hinreichend flexibel und individuell gestaltbar sein, um persönliche Vorlieben und Wünsche des einzelnen Benutzers berücksichtigen zu können.

Der zweite Schwerpunkt der Arbeit liegt auf einer Performanzsteigerung im Bereich der Anfragebearbeitung. Prämisse der Arbeit ist, dass durch eine vollständige Integration der Dienste die vorhandene Informationsfülle gewinnbringend genutzt und so dem Benutzer der größte Mehrwert geboten werden kann. Dafür soll in der Arbeit nun untersucht werden, mit welchen technischen Möglichkeiten und Verfahren eine Verbesserung erzielt werden kann. Die zugrundeliegenden Dienstparameter sind gewissermaßen vorgegeben und dienen als Basis für die folgenden Gestaltungs- und Optimierungsüberlegungen. Dabei soll schließlich quantitativ belegt werden, in welchem Maße und unter welchen Umständen und Randbedingungen eine Performanzsteigerung möglich ist.

Damit zielt die Arbeit in erster Linie darauf ab, die Literaturrecherche für den Benutzer in der Praxis zu verbessern und die Potentiale und Synergien im bestehenden Dienstangebot zu

identifizieren und möglichst gut und geeignet nutzbar zu machen. Um „den“ Benutzer bei der Literaturrecherche zu konkretisieren, in seinen Ansprüchen, Wünschen und auch in seinen Fertigkeiten, sollen Ergebnisse von zahlreichen Benutzerstudien herangezogen werden.

1.4 Gliederung

Die Arbeit gliedert sich wie folgt. Kapitel 2 führt zunächst das Szenario der wissenschaftlichen Literaturrecherche ein und formuliert die Anforderungen an ein Meta-Recherchesystem aus Benutzer-, Anbieter- und aus technischer Sicht. Kapitel 3 gibt eine Übersicht über den Stand der Technik und betrachtet einerseits existierende Systeme und Ansätze, andererseits in der Literatur vorgeschlagene Konzepte und Techniken, die zur Konkretisierung und Umsetzung der Anforderungen berücksichtigt werden sollen. Kapitel 4 stellt die zentrale Idee des Lösungsansatzes vor und gibt einen Überblick über die konzipierte Gesamtlösung, die in den folgenden Kapiteln detaillierter beschrieben wird.

Kapitel 5-7 behandeln die Konzeption einer geeigneten Benutzungsschnittstelle, d.h. der Teile des Meta-Recherchesystems, die für den Benutzer sichtbar sind und die die Interaktion mit dem System beeinflussen. Kapitel 8-11 betreffen die Konzeption einer geeigneten internen Anfragebearbeitung im Hinblick auf eine Performanzsteigerung des Gesamtsystems. Kapitel 12-13 beschreiben die Implementierung und die Validierung des Ansatzes.

Konzeption Rechercheschnittstelle: In Kapitel 5 wird zunächst ein Informationsmodell zur einheitlichen Repräsentation der Dokumente entworfen, welches die Wahrnehmung durch den Benutzer berücksichtigt. Im Gegensatz zu Meta-Suchmaschinen für das WWW wird für die Integration von Literaturkatalogen ein explizites Domänenmodell benötigt. In Kapitel 6 wird eine hinreichend mächtige Anfragesprache für ein Meta-Recherchesystem zur Literatursuche entwickelt. In Kapitel 7 wird das Konzept der generischen Suchmuster eingeführt, welches sowohl Gewohnheiten und Fertigkeiten eines Benutzers berücksichtigt, als auch der Informationsflut durch präzise Anfragemechanismen entgegenwirken kann. Darüber hinaus wird eine geeignete Rechercheschnittstelle für die zuvor entwickelte Anfragesprache und das Konzept der generischen Suchmuster entworfen.

Konzeption Anfragebearbeitung: Zunächst werden die Grundlagen und Rahmenbedingungen erarbeitet und zusammengestellt, auf denen die Anfragebearbeitung stattfinden kann. Dazu werden in Kapitel 8 die strukturellen Eigenschaften und das Antwortzeitverhalten existierender Literaturdienste genauer untersucht. In Kapitel 9 wird ein modularer Lösungsansatz für die Anfragebearbeitung entwickelt, welcher eine Reihe von unterschiedlichen Anfragebearbeitungsstrategien enthält. Kapitel 10 beschäftigt sich mit einem speziellen Teil der Anfragebearbeitung, der Duplikaterkennung. Diese ist zentral wichtig für die Hervorbringung der genannten Synergieeffekte. Um die verschiedenen Strategien der Anfragebearbeitung miteinander vergleichen und bewerten zu können, werden Maße für die Bewertung der Performanz des Systems benötigt. Diese werden in Kapitel 11 definiert.

Implementierung und Validierung: Kapitel 12 stellt das entwickelte Simulationswerkzeug vor, mit welchem die Performanzuntersuchungen durchgeführt werden und die Validierung des vorgeschlagenen Konzepts erfolgt. Kapitel 13 dokumentiert die durchgeführten Experimente und formuliert daraus die Ergebnisse der Arbeit.

Schließlich fasst Kapitel 14 die Arbeit zusammen und gibt einen Ausblick auf weitere Einsatzmöglichkeiten und zukünftige Entwicklungen.

2 Anforderungsanalyse

2.1 Übersicht

In diesem Kapitel werden die Anforderungen an Meta-Recherchesysteme zur Unterstützung der wissenschaftlichen Literaturrecherche analysiert. Zunächst werden anhand eines Szenarios das Umfeld, die aktuelle Situation und einige zentrale Fragestellungen illustriert. Dazu werden verschiedene Arten von Recherchediensten sowie charakteristische Benutzeranfragen vorgestellt. Abschnitt 2.3 ermittelt aus dem vorgestellten Szenario die Anforderungen. Ergebnis dieses Abschnitts sind eine Reihe von Kriterien zur Bewertung bereits existierender Lösungsansätze. Abschnitt 2.4 führt schließlich eine Architektur ein, die die Erfüllung aller Kriterien zum Ziel hat und die dieser Arbeit zugrundegelegt wird. Der letzte Abschnitt (Abschnitt 2.5) hebt die Teile der Architektur hervor, die Gegenstand der vorliegenden Arbeit sind. Dazu werden die Ziele der Arbeit festgelegt und die Anforderungen, denen eine Lösung genügen soll, erläutert.

2.2 Szenario

Als Szenario für die Arbeit wird die wissenschaftliche Literaturrecherche gewählt. Sie stellt eine spezielle Form der Informationssuche dar: Die Literaturrecherche soll der beruflichen Arbeit dienen. Es wird erwartet, dass sie sorgfältig und systematisch durchgeführt wird. Im Gegensatz dazu werden keine speziellen Erwartungen an die allgemeine Informationssuche gestellt, diese kann oft einen spontanen und eher unterhaltenden Charakter haben. Bei den gesuchten Informationen einer wissenschaftlichen Literaturrecherche handelt es sich i.d.R. um Veröffentlichungen, z.B. um Bücher, Zeitschriften oder Konferenzartikel. Eine große Bedeutung kommt in der Forschung auch der sogenannten grauen Literatur zu. Als solche bezeichnet man (noch) nicht veröffentlichte Dokumente, z.B. Vorabdrucke (Preprints) oder Technische Berichte.

Es gibt zahlreiche Recherchedienste, bei denen sich ein Benutzer zu einem Thema über existierende Dokumente informieren kann. Zur Beurteilung eines Dokuments kann er nicht nur die bibliographischen Daten abfragen. Bei einigen Diensten sind Zusatzinformationen verfügbar, z.B. das Inhaltsverzeichnis eines Buches, das Titelbild, eine Inhaltsangabe, Leserbewertungen, Buchbesprechungen, eine Leseprobe und möglicherweise sogar der Volltext. Auch Informationen zu Preisen und Lieferzeiten bei Buchhändlern oder Ausleihmöglichkeiten bei Bibliotheken sind abrufbar.

Ein Informationswunsch eines Studenten in der Anfangsphase seiner Diplomarbeit könnte beispielsweise lauten: *"Ich suche ein Lehrbuch zum Thema Java. Das Buch soll relativ aktuell sein [..., damit es die Neuerungen der Java-Version 1.3 oder später abdeckt]. Es soll lieber in deutscher Sprache sein, unter Umständen ist auch englisch akzeptabel. Am liebsten würde ich das Buch in einer Bibliothek vor Ort [hier: Karlsruhe] ausleihen. Da ich das Buch allerdings möglichst bald benötige [und vielleicht auch länger behalten und mit persönlichen Notizen versehen will], würde ich es mir möglicherweise auch selbst kaufen."* Zur Befriedigung eines derartigen Informationsbedürfnisses sollte ein Meta-Recherchesystem mehrere Bibliotheken vor Ort beinhalten, aber auch mindestens einen Online-Buchhändler für Deutschland.

Der Abschnitt ist in drei Unterabschnitte aufgeteilt: Zuerst werden die für das Szenario benötigten Literaturrecherche- und -beschaffungsdienste beschrieben. Anschließend werden mögliche Arten von Anfragen an das System vorgestellt. Und schließlich wird erläutert, wie die Beantwortung von Anfragen an ein Meta-Recherchesystem in etwa aussehen könnte.

2.2.1 Dienste für eine wissenschaftliche Literaturrecherche

Im Bereich der wissenschaftlichen Literaturrecherche kann man verschiedene Klassen von Diensten identifizieren [SO02a]. Angesichts der verfügbaren Funktionalität und der bereitgestellten Informationen weisen diese Klassen jeweils charakteristische Merkmale auf. In einem umfassenden und repräsentativen Meta-Recherchesystem sollten möglichst viele dieser verschiedenen Dienstklassen vertreten sein. Durch das Zusammenführen der Dienste können nämlich Synergieeffekte entstehen und genutzt werden, d.h. die unterschiedlichen Stärken der Dienste können sich optimal ergänzen und etwaige Schwächen fallen dadurch nicht mehr ins Gewicht. Im Folgenden werden nun die im Bereich der Informatik relevanten Literaturrecherche- und -beschaffungsdienste mit ihren Vorzügen und Nachteilen erläutert. In Anhang A sind zu den angegebenen Diensten die vollständigen Namen und Webadressen aufgeführt.

Wissenschaftliche Bibliotheken. Sie bieten ihrem Kundenkreis vor Ort einen umfassenden und regelmäßig aktualisierten Bestand an Büchern und Zeitschriften zum Recherchieren, Einsehen und Ausleihen an. Die Ausleihoption ist für Viel-Leser besonders nützlich, leider sind die neuesten und beliebtesten Werke häufig über lange Zeiträume hinweg vergriffen. Der geführte Dokumentbestand wird von Fachpersonal ausgewählt, geordnet und auf die Bedürfnisse der Benutzer vor Ort abgestimmt. Für die Benutzung einer Bibliothek (zumindest für das Ausleihen von Büchern) ist ein Benutzerausweis und das persönliche Erscheinen vor Ort notwendig. Im Rahmen eines Meta-Recherchesystems könnte ein Benutzer zwar von der Recherchefunktionalität von beliebigen Bibliotheken Gebrauch machen, die Ausleihfunktion kann er allerdings nur bei Bibliotheken vor Ort nutzen. In Karlsruhe gibt es z.B. die Universitätsbibliothek (UBKA) und die Badische Landesbibliothek (BLB).

Fachdatenbanken. Fachinformationszentren betreiben und pflegen eine Reihe von Fachdatenbanken. Diese bieten einen vollständigen Nachweisdienst zu bestimmten Fachgebieten an. In Fachdatenbanken werden sämtliche veröffentlichten Dokumente katalogisiert und mit allen bibliographischen Daten inklusive einer Zusammenfassung archiviert. Leider bieten derartige Dienste keine Möglichkeit an, die nachgewiesenen Dokumente auch zu beschaffen. Meistens ist bereits die Recherche in Fachdatenbanken kostenpflichtig, wobei eine halbstündige Recherche leicht einige Hundert Euro kosten kann. Die für den Bereich der Informatik interessanten Fachdatenbanken sind CompuScience, INSPEC und ITEC.

Dokumentenlieferdienste. Da Bibliotheken nur einen Teil aller erscheinenden Zeitschriften vorhalten können, werden zusätzlich Dokumentenlieferdienste benötigt. Diese liefern dem Benutzer in der Regel Kopien von Zeitschriftenartikeln direkt an seinen Arbeitsplatz, sei es in elektronischer Form, per Fax oder über den herkömmlichen Postweg. Teilweise liegen die Artikel bereits elektronisch vor, teilweise werden sie zuerst eingescannt. Die Bestellung eines Artikels kostet etwa zwischen 4 und 10 Euro. Das bekannteste Beispiel ist der deutschlandweite Dokumentenlieferdienst Subito. Über ihn kann jeder beliebige Artikel bezogen werden, sofern er in *einer* Bibliothek in Deutschland vorhanden ist. Ein weiteres Beispiel ist LEA, ein Dienst der Universitätsbibliothek Karlsruhe, welcher den Universitätsangehörigen die in der Bibliothek vorhandenen Artikel auch in elektronischer Form an den Arbeitsplatz liefert.

Fachgesellschaften. Fachgesellschaften sind häufig auch als Herausgeber und Verleger von zahlreichen Zeitschriften und Konferenzbänden tätig. Mit den elektronischen Volltexten ihrer Materialien haben sie in den letzten Jahren eigene Digitale Bibliotheken aufgebaut. Diese sind meist nur für die Mitglieder der Fachgesellschaft und gegen einen geringen Aufpreis zum Mitgliedsbeitrag zugänglich. Im Bereich der Informatik sind beispielsweise die Digitalen Bibliotheken von ACM und IEEE sehr bekannt.

Buchhandel. Online-Buchhändler zeichnen sich durch ihren großen Bestand an Dokumenten, meist Büchern, aus. Sie enthalten nahezu sämtliche in Deutschland lieferbaren Publikationen. Über Online-Buchhändler kann sich der Benutzer jedes Dokument bequem nach Hause oder an seinen Arbeitsplatz liefern lassen. Zu einem Dokument werden zahlreiche, für den Benutzer wertvolle Zusatzinformationen angeboten. Neben den bibliographischen Daten werden beispielsweise Zusammenfassungen, Titelbilder, Leserbewertungen und der aktuelle Verkaufsrang zur Verfügung gestellt. Weiterhin wird jeweils der Preis und die Lieferfrist angegeben. Bekannte Online-Buchhändler in Deutschland sind amazon.de (AMAZON) und buch.de (BUCH).

Volltextarchive. Elektronische Volltextarchive sind im wissenschaftlichen Bereich häufig anzutreffen, sie enthalten meist graue Literatur, d.h. Technische Berichte oder Vorabdrucke,

die von interessierten Benutzern direkt und kostenfrei heruntergeladen werden können. Ein Beispiel im Bereich der Informatik ist NCSTRL.

Dienstangebote von Privatpersonen. Im wissenschaftlichen Bereich findet man immer wieder Dienste, die – einst von engagierten Privatpersonen ins Leben gerufen – auch weiterhin kontinuierlich gepflegt und verbessert werden. Diese hochwertigen Dienste werden häufig an universitären Einrichtungen betrieben, werden schnell weltweit bekannt und sind von der Forschungsgemeinschaft hoch geschätzt. Ein Beispiel ist die Informatik-Bibliographie (CSBIB), welche den wohl umfassendsten Bestand an Informatik-Artikeln beinhaltet, teilweise sogar im Volltext. Ein weiteres Beispiel ist DBLP.

Forschungsprototypen. Im Forschungsumfeld von Digitalen Bibliotheken sind in den letzten Jahren einige Prototypen entstanden, die mittlerweile aufgrund ihrer guten Qualität und Leistung als eigenständige Dienste betrieben werden. Beispiele hierfür sind CORA und ResearchIndex. Beide Dienste sammeln automatisch verfügbare Online-Versionen von wissenschaftlichen Artikeln im World-Wide Web. ResearchIndex wertet zudem zwischen den einzelnen Dokumenten noch die Zitiert-Verweise aus, welche in der Wissenschaft bekanntlich eine wichtige Rolle spielen.

Die gemachten Ausführungen zeigen, wie reichhaltig und vielfältig das Dienstangebot ist, welches ein Benutzer zur Zeit im World-Wide Web vorfindet. Es wurden lediglich einige Vertreter von verschiedenen Dienstklassen zur wissenschaftlichen Literaturrecherche vorgestellt. Zusätzlich zur Anzahl und Vielfalt zeichnet sich das Dienstangebot durch eine große Dynamik aus, d.h. immer wieder entstehen neue Dienste und die bestehenden Dienste werden weiter entwickelt.

Im Wesentlichen unterscheiden sich die vorgestellten Recherchedienste in drei Bereichen: im nachgewiesenen Dokumentbestand, in der Menge der Informationen, die zu einem Dokument abrufbar sind, und in der Funktionalität zum Beschaffen von Dokumenten. Betrachtet man die Synergieeffekte, die durch die Kombination der vorhandenen Dienste möglich werden, so sind diese beachtlich: Es können nahezu alle denkbaren Dokumente aufgespürt werden, aus beliebigen Fachrichtungen und unabhängig vom Publikationstyp (Buch, Zeitschrift, Artikel). An Zusatzinformationen, die dem Benutzer die Entscheidung erleichtern, mangelt es auch nicht. Und wenn der Benutzer sich schließlich ein Dokument beschaffen will, kann er es entweder ausleihen, sich zuschicken lassen oder vielleicht auch sofort den elektronischen Volltext beziehen.

In der technischen Realisierung sind sich die vorgestellten Recherchedienste hingegen recht ähnlich. Sie sind alle über eine Schnittstelle über das World-Wide Web zugänglich und arbeiten nach demselben Prinzip: Der Benutzer trägt seine Anfrage in Form von Stichworten in einem Formular mit verschiedenen Feldern ein, z.B. bei Titel oder Autor. Die Ergebnisse werden von den Diensten daraufhin ermittelt und portionsweise zurückgeliefert. Zunächst wird dem Benutzer eine HTML-Seite mit einer Kurztitelliste präsentiert, welche zu den

einzelnen Dokumenten meist nur die wichtigsten Informationen wie Titel, Autor und Jahr enthält. Bei Bedarf kann der Benutzer anschließend sämtliche verfügbaren Informationen zu einem Dokument abrufen, indem er die entsprechenden Verweise zu weiteren HTML-Seiten mit eben diesen Informationen verfolgt.

Auf der syntaktischen und semantischen Ebene weisen die einzelnen Dienste wieder deutliche Unterschiede auf: Die Anfragesprachen unterscheiden sich hinsichtlich der Menge der Felder, nach denen der Bestand durchsucht werden kann, sowie hinsichtlich der Verknüpfungsmöglichkeiten zwischen den einzelnen Teilanfragen. Auch die Bezeichnungen der Felder sind nicht einheitlich gewählt. Die deutlichsten Unterschiede zeigen sich allerdings in den Ergebnisdarstellungen: sowohl Datenformate als auch Dateninhalte weisen eine große Heterogenität auf.

2.2.2 Mögliche Anfragen und Suchstrategien

Ein Benutzer kann im Laufe seiner wissenschaftlichen Arbeit z.B. mit folgenden Arten von Fragen an ein Meta-Recherchesystem herantreten:

- *Ich habe ein Buch empfohlen bekommen, "Digitale Bibliotheken" von A. Endres und D. Fellner. Nun möchte ich mich darüber genauer informieren, zum Inhalt und darüber, wo ich es ggf. bekommen kann.*

Eine derartige Anfrage bezeichnet man als *ready reference*. Hier sollen zu einem bestimmten Dokument möglichst alle verfügbaren Informationen automatisch zusammengetragen werden. Eine angemessene Anfragebearbeitung würde beispielsweise zuerst die vollständigen bibliographischen Daten zusammentragen, also Titel, Untertitel, Erscheinungsdatum, Verlag, Seitenanzahl, ISBN, usw. Um weitere Informationen über den Inhalt zu ergänzen, können Buchkataloge wie amazon.de oder der entsprechende Verlagskatalog angefragt werden. Dort erhält man Zusammenfassungen, Kurzbeschreibungen, das Inhaltsverzeichnis sowie ein Foto vom Titelbild. Weiterhin interessieren den Benutzer Beschaffungsinformationen. Dazu kann man bei den einzelnen Bibliotheken vor Ort anfragen, ob das Buch vorhanden ist und wenn ja, ab wann es ausleihbar ist. Vergleichsweise kann man den Preis und die Lieferdauer bei einem Buchhändler angeben.

- *Ich möchte mir möglichst schnell ein gutes Lehrbuch über Java ausleihen.*

In dieser Anfrage sind einige Bedingungen explizit und andere implizit formuliert. Es sollen also nur Bibliothekskataloge berücksichtigt werden, für die der Benutzer auch eine Zugangsberechtigung hat. Das sollte das System für angemessene Entscheidungen auch wissen. In diesem Fall seien es die Bibliotheken vor Ort, also UBKA und BLB. Als nächstes ist die Verfügbarkeit entscheidend. Also müssen noch vor weiteren inhaltlichen Informationen die Ausleihzeiten erfragt werden. Da mit dem Stichwort "Java" eine große Ergebnismenge zu erwarten ist, der Benutzer letztlich aber nur ein einziges Buch bekommen möchte, ist es sinnvoll, in die Suchstrategie noch weitere Kriterien mit einzubeziehen:

Beispielsweise sind aktuelle Bücher in der Regel interessanter. Oder über den Benutzer ist bekannt, dass er zwar sowohl deutsch als auch englisch als Sprache beherrscht, ihm aber ein deutsches Buch unter gleichen Bedingungen wesentlich angenehmer ist. Weiterhin wird die UBKA aufgrund der besseren Erreichbarkeit vor Ort (sie befindet sich direkt auf dem Campus) bevorzugt. Eine Anfragebearbeitung sollte dabei keine Dokumente ausschließen, könnte aber durch eine entsprechende Sortierreihenfolge dem Benutzer die Entscheidung erleichtern.

- *Ich möchte einen Fachartikel, der den Stand der Forschung im Bereich Benutzermodellierung möglichst überblicksartig oder einführend darstellt. Den Artikel möchte ich mir dann auch gleich ausdrucken.*

Bei dieser Anfrage können die zu befragenden Quellen aufgrund der gewünschten Dokumentart ausgewählt werden. Nur Dienste, die Artikel beinhalten und dabei auch Volltexte zur Verfügung stellen, kommen in Frage. Die Anfrage kann zum Beispiel um Stichworte ergänzt werden, die die Ausrichtung des Artikels noch näher beschreiben: z.B. Überblick, Einführung, Stand der Forschung oder – da die meisten Artikel in englischer Sprache geschrieben sind – entsprechend survey, introduction, state of the art.

- *Was hat der Forscher N. Belkin eigentlich alles publiziert und welches waren seine erfolgreichsten Publikationen?*

Bei dieser Anfrage will sich der Benutzer einen Überblick über eine Menge von existierenden Werken verschaffen. Seine Absicht ist es nicht, sich – wie in den vorangegangenen Beispielen – schließlich für ein Dokument entscheiden und dieses dann auch beschaffen zu wollen. Im vorliegenden Fall sind ausführliche Zusatzinformationen oder gar konkrete Beschaffungsinformationen zweitrangig, vielmehr wird ein vollständiger Überblick angestrebt. Eine Anfragebearbeitung sollte zuerst die Dienste zu befragen, die große und umfassende Dokumentbestände nachweisen können, wie z.B. die Informatik-Bibliographie (CSBIB) oder eine Fachdatenbank wie INSPEC. Zusätzlich können auch andere Dienste befragt werden. In jedem Fall ist bei der Präsentation des Gesamtergebnisses eine Duplikateliminierung wichtig; eine chronologische Ordnung in Bezug auf das Erscheinungsjahr wäre sicherlich auch hilfreich. Für die Beurteilung der Bedeutung einer Veröffentlichung kann ein weiterer Dienst befragt werden, der Informationen zu Zitierungshäufigkeiten bereitstellt, z.B. ResearchIndex oder der Science Citation Index (SCI).

2.2.3 Beantwortung von Anfragen

Um derartige Anfragen beantworten zu können, sind mehrere Schritte erforderlich. Zunächst müssen die geeigneten Dienste identifiziert werden, sofern sie nicht explizit in der Anfrage angegeben oder direkt ableitbar sind. Dazu ist die Kenntnis aller verfügbaren Dienste und ihrer Dienstmerkmale notwendig, z.B. in Bezug auf ihren Dokumentbestand, ihr Informationspotential und ihre Beschaffungsfunktionalitäten.

Die ausgewählten Recherchedienste können nun parallel oder in einer geeigneten Reihenfolge gemäß ihrer Anfragemöglichkeiten befragt werden. Die Anfragen müssen in der Syntax und Semantik des jeweiligen Dienstes formuliert sein, die Ergebnisse müssen aus dem Format des Dienstes in ein einheitliches Format für die interne Weiterverarbeitung übersetzt werden. Teilweise können die Bedingungen des Benutzers in den Anfragen an die Dienste mitübergeben werden, teilweise müssen die Bedingungen allerdings auch über nachträgliche Filteroperationen durchgesetzt werden, insbesondere bei Bedingungen, die die Sprache, den Preis oder die Ausleihbarkeit betreffen, da diese Informationen bei den Diensten in der Regel nicht direkt anfragbar sind.

Die Anfragebearbeitung ist als äußerst zeitkritisch einzustufen, da der Benutzer interaktiv auf die Ergebnisse wartet. Gerade wenn die zugrundeliegenden Dienste über Webschnittstellen angesprochen werden, kann die Antwortzeit lange dauern, wenn größere Ergebnismengen zu erwarten sind und diese mit vollständigen Informationen zusammengestellt werden sollen. Zum Beschaffen dieser Informationsstücke sind jeweils separate Web-Anfragen notwendig und deren Antwortzeiten liegen, grob und pauschal gesprochen, im Bereich zwischen 1 und 4 Sekunden (vgl. Kapitel 8 bzw. [SO02a]). Benutzer sind – was Wartezeiten betrifft – extrem ungeduldig. Daher ist bei der Anfragebearbeitung, neben der Überwindung der semantischen Heterogenität, besonders auf Performanz zu achten. Natürlich kann ein Meta-Recherchesystem die Antwortzeiten der zugrundeliegenden Dienste selbst nicht beeinflussen, aber es kann beispielsweise Anfragen geschickt umformulieren und Anfragepläne generieren, die weniger Web-Anfragen benötigen.

Schließlich müssen die gefundenen Informationen zusammengeführt und integriert werden. Dabei müssen z.B. Duplikate erkannt werden. Die Gesamtergebnismenge soll dann dem Benutzer möglichst rasch und in einer angemessenen Form präsentiert werden. Auch die Duplikaterkennung und -eliminierung auf den eingesammelten Ergebnissen sollte durch schnelle Algorithmen umgesetzt werden, um zumindest keine unnötigen zusätzlichen Wartezeiten für den Benutzer zu erzeugen.

2.3 Anforderungen

In diesem Abschnitt wird das oben eingeführte Szenario analysiert. Ziel dieser Analyse ist es, die wesentlichen Anforderungen an ein Meta-Recherchesystem zur wissenschaftlichen Literaturrecherche und -beschaffung herauszuarbeiten.

In einem ersten Schritt ist es wichtig, die vorhandenen Dienste in ein Gesamtsystem zu integrieren. Im zweiten Schritt soll dieses komplexe System auf die Bedürfnisse des Benutzers zugeschnitten und für ihn überschaubar und handhabbar gemacht werden.

Die Vorgehensweise ist wie folgt: Zuerst werden die Anforderungen an ein solches Meta-Recherchesystem sowohl von Seiten der Informationsanbieter als auch von Seiten des

Benutzers her zusammengestellt. Daraus werden anschließend die konkreten, eher technisch orientierten Anforderungen an das gesamte Recherchesystem abgeleitet, welche der Gegenstand der vorliegenden Arbeit sind.

2.3.1 Anforderungen der Dienstanbieter

Autonomie der Recherchedienste. Im Literaturrechercheszenario ist es wichtig, die Autonomie der Dienste zu bewahren. Jeder einzelne Recherchedienst verwaltet und pflegt seinen eigenen Dokumenten- und Datenbestand. Die Anbindung eines Dienstes an ein verteiltes Recherchesystem darf keine Änderungen am Dienst selbst erfordern, da dieser ja weiterhin auch als eigenständiger Dienst betrieben werden soll.

Lastvorgaben der Recherchedienste. Die Teilnahme an einem Meta-Recherchesystem bedeutet für einen Dienst nicht nur eine größere Öffentlichkeitswirksamkeit und mehr Benutzer, sondern auch einen vermehrten Ressourcenverbrauch. Da der Dienst dem ursprünglichen Kundenkreis weiterhin in seiner gewohnten Qualität und Performanz zur Verfügung stehen soll, muss es möglich sein, Rahmenbedingungen der Dienste hinsichtlich einer akzeptablen Zusatzbelastung durch Anfragen des Meta-Recherchesystems zu berücksichtigen.

2.3.2 Anforderungen des Benutzers

Umfassendes Dienstangebot. Der Benutzer ist an einem großen, umfassenden und aktuellen Dienstangebot interessiert. Daher ist es wichtig, dass viele existierende Dienste in das System eingebunden werden können. Diese Forderung kommt eher von der Benutzerseite her. So muss der Benutzer die einzelnen Dienste nicht mehr ausdrücklich kennen und kann sich leicht einen Überblick über die bestehenden Möglichkeiten verschaffen. Die Dienstanbieter selbst sind meist nicht an einer direkten Vergleichbarkeit von ihren Angeboten mit denen der Konkurrenz interessiert.

Integrierte Nutzung der Dienste durch einheitlichen Zugriff. Für den Benutzer stellt eine einheitliche Zugriffsmöglichkeit auf die einzelnen Recherchedienste eine erhebliche Erleichterung dar: So braucht er sich nur eine Anfragesprache einzuprägen und kann sich an eine Präsentationsform der Ergebnisse gewöhnen. Die integrierte Nutzung erlaubt es dem Benutzer, über einen zentralen Zugangspunkt mehrere parallele Einzelanfragen abzusetzen und dadurch ein größeres Angebot zu erreichen.

Einfache Handhabbarkeit. Diese Forderung ist prinzipiell bei jedem System mit direktem Benutzerkontakt zu berücksichtigen. Besonders wichtig ist sie für neue und ungeübte Benutzer sowie für Benutzer, die eher selten Kontakt mit dem System haben. Weist das System zudem noch eine relativ komplexe Funktionalität auf, wie das beispielsweise bei einem Meta-Recherchesystem der Fall ist, so kommt dieser Anforderung eine entscheidende Bedeutung zu.

Ausdrucksstarke Anfragemöglichkeiten. Gerade in einem Meta-Recherchesystem sind große Ergebnismengen zu erwarten. Um der Informationsüberflutung schon am Ursprung ihrer Entstehung entgegenwirken zu können, sind ausdrucksstarke Anfragemechanismen und Formulierungsmöglichkeiten vorzusehen, zumindest für den erfahrenen und versierten Nutzerkreis.

Aussagekräftige Ergebnispräsentation. Ebenfalls bei großen Treffermengen ist eine prägnante Darstellungsform für die Ergebnisdokumente zu wählen. Die Darstellung soll dem Benutzer einerseits einen guten Überblick über die Gesamtergebnismenge ermöglichen. Andererseits soll sie auch ein detaillierteres Studieren und letztendlich eine Entscheidungsfindung unterstützen.

Kurze Wartezeiten. Auch diese Forderung ist beim vorliegenden Szenario – genauso wie bei jeder anderen interaktiven Anwendung – entscheidend für die Benutzerakzeptanz und damit für den Erfolg des Gesamtsystems.

2.3.3 Anforderungen an das Meta-Recherchesystem

Aus den Anforderungen der Dienstanbieter und denen des Benutzers ergeben sich nun die Anforderungen an das Meta-Recherchesystem, sowohl was die Funktionalität einzelner Systemkomponenten als auch was die Benutzungsoberfläche anbelangt. In diesem Abschnitt werden die vorher genannten Anforderungen also konkretisiert und eher technisch formuliert.

Autonomie der Recherchedienste.

- **Kein Eingriff in die Recherchedienste.** Das Recherchesystem soll auf die zugrundliegenden Dienste nur über deren HTTP-Schnittstelle zugreifen. Damit stehen dem Meta-Recherchesystem genau dieselben Aktionen und Informationen zur Verfügung, die ein Benutzer bei einer manuellen Recherche im World-Wide Web bei den entsprechenden Diensten auch verwenden könnte.
- **Evolution der Recherchedienste.** Gerade wenn man auf die zugrundeliegenden Recherchedienste über ihre Schnittstelle im World-Wide Web zugreift, muss man damit rechnen, dass sich das Erscheinungsbild der entsprechenden HTML-Seiten häufig ändern wird. Diese Änderungen betreffen allerdings hauptsächlich die graphische Darstellung und Anordnung der Informationen, die wesentlichen Merkmale sowie die Funktionalität der Dienste sind davon im Allgemeinen nicht betroffen.

Lastvorgaben der Recherchedienste.

- **Lastregulierende Anfragebearbeitung.** Den zugrundeliegenden Recherchediensten steht eine bestimmte Menge an Ressourcen zur Verfügung. Daher sind sie meist nur gewillt, an einem Meta-Recherchesystem teilzunehmen, sofern sich die dadurch zusätzlich entstehende Last an Web-Anfragen und Netzverkehr in akzeptablen Bereichen bewegt. Daher ist für das Meta-Recherchesystem eine Anfragebearbeitung

notwendig, die die erzeugte Last kontrollieren und steuern kann. Dazu kann die Anfrageplanung beispielsweise kritische Aktionen ablehnen, zeitlich verzögern oder auf andere Alternativen ausweichen.

Umfassendes Dienstangebot.

- **Offenheit.** Um die Menge der integrierten Recherchedienste möglichst umfassend und repräsentativ gestalten zu können, sollen möglichst beliebige Literaturrecherchedienste integriert werden können. Als einzige Beschränkung wurde weiter oben die Zugänglichkeit über eine HTTP-Schnittstelle gefordert, die allerdings mittlerweile von nahezu allen Diensten zur Verfügung gestellt wird. Ansonsten soll das Recherchesystem in der Lage sein, mit unterschiedlich strukturierten und präsentierten Inhalten umgehen zu können.
- **Dynamik.** Das Dienstangebot im Bereich der Literaturrecherche und -beschaffung ist sehr dynamisch. Schnell entstehen neue Dienste und genauso schnell sind oft auch bestehende Dienste verschwunden, die dem Marktgeschehen nicht standhalten konnten. Daher muss das Meta-Recherchesystem Mechanismen für einfache Zu- und Abgänge von diesen Diensten vorsehen.
- **Skalierbarkeit.** Das Dienstangebot im Bereich der Literaturrecherche und -beschaffung ist sehr reichhaltig. Es gibt verschiedene Klassen von Diensten und innerhalb dieser Klassen zahlreiche Vertreter. Daher kann die Zahl der integrierten Dienste recht groß werden. Das Meta-Recherchesystem muss Möglichkeiten vorsehen, auch mit einer großen Menge von Diensten umgehen zu können.

Integrierte Nutzung der Dienste durch einheitlichen Zugriff.

- **Einheitliche Anfragesprache.** Die Forderung des Benutzers nach einem einheitlichen Zugriff auf die Recherchedienste bedeutet für das Meta-Recherchesystem, dass auf die Informationen unabhängig davon, über welchen Dienst sie auch immer bezogen werden, einheitlich zugegriffen werden kann. Dazu muss das System eine einheitliche Anfragesprache zur Verfügung stellen.
- **Einheitliche Ergebnisdarstellung.** Genauso muss das Meta-Recherchesystem die unterschiedlichen Teilergebnisse der einzelnen Dienste in einer einheitlichen Form präsentieren. Dazu müssen die einzelnen Datensätze in ein einheitliches Modell, ein sogenanntes Domänenmodell, gebracht werden. Durch den Einsatz eines Domänenmodells wird die syntaktische und semantische Heterogenität der Datensätze überwunden, die interne Weiterverarbeitung und schließlich die einheitliche Ergebnisdarstellung ermöglicht.
- **Transformation von Benutzeranfragen und Einzelergebnissen.** Um die vorangegangenen zwei Forderungen erfüllen zu können, sind zwei Arten von Transformationen notwendig, da wir es im Literaturrechercheszenario mit heterogenen Basisdiensten zu tun haben, was die Anfragemöglichkeiten und die Ergebnispräsenta-

tionen betrifft. Eine Benutzeranfrage muss also in die Syntax und Semantik der Anfragesprachen der zu befragenden Dienste übersetzt werden, damit diese Dienste die Anfragen korrekt bearbeiten können. Die Ergebnisse müssen wiederum im gelieferten Format, d.h. auch hier in der entsprechenden Syntax und Semantik, entgegengenommen werden und anschließend in ein geeignetes einheitliches Format für die interne Weiterverarbeitung übersetzt werden.

- **Koordinierung von Anfragen und Ergebnissen.** Zur Bearbeitung einer Benutzeranfrage ist es häufig notwendig, eine Reihe von Einzelabfragen an die zugrundeliegenden Dienste abzusetzen und deren Ergebnisse dann in geeigneter Weise zu kombinieren. Um das gewünschte Gesamtergebnis zu erreichen, sind Strategien für den Umgang mit diesen Einzelaufrufen zu entwickeln. Aufgrund von Datenabhängigkeiten und der technischen Realisierung der Recherchedienste können Aufrufe nicht immer in beliebiger Reihenfolge ausgeführt werden.
- **Duplikatbehandlung.** Um die Synergien der Dienste auf Informationsebene nutzen zu können, sind Verfahren zur Duplikaterkennung und zur anschließenden Eliminierung oder Verschmelzung der Dokumentreferenzen bereitzustellen.

Angemessene Benutzerinteraktion.

- **Einfache Handhabbarkeit.** Um dem Benutzer die Interaktion mit dem System zu erleichtern, soll er auch beim Meta-Recherchesystem die bekannten Standard-Techniken für die Realisierung von Benutzungsoberflächen vorfinden, z.B. formularbasierte Anfragemöglichkeiten, Funktionsleisten, Kontextmenüs oder Auswahlknöpfe. Auf Experimente mit neuartigen Anfrage- bzw. Ergebnisparadigmen soll im Rahmen dieser Arbeit verzichtet werden.
- **Kontrollmöglichkeit des Benutzers.** Bei der Gestaltung von Benutzungsschnittstellen gelten Kompetenzförderlichkeit und Handlungsflexibilität als zentrale Grundprinzipien. Bei einem integrierten Recherchesystem treten häufig große Informationsmengen auf und damit auch zahlreiche Freiheitsgrade im Umgang mit diesen Informationen. Dazu soll der Benutzer zum einen über ausdrucksstarke Anfragemöglichkeiten und über die Formulierung von Suchstrategien in die Lage versetzt werden, sich sein gewünschtes Ergebnis individuell zusammenstellen und gestalten zu können. Zum anderen sollen flexible Nachbearbeitungsoperatoren dem Benutzer eine aussagekräftige und auf seine persönlichen Bedürfnisse abgestimmte Ergebnispräsentation ermöglichen.
- **Berücksichtigung menschlicher Eigenschaften.** Bei der Gestaltung interaktiver Systeme sollten die physiologischen und kognitiven Fähigkeiten und Grenzen eines Benutzers berücksichtigt werden. Das betrifft hier die zu erwartenden Fertigkeiten im Bereich Anfrageformulierung und Ergebnisbegutachtung. Weiterhin sollten Benutzerstudien im Anwendungsumfeld hinzugezogen werden, beispielsweise

Beobachtungen typischer Vorgehensweisen und Verhaltensmuster bei einer Informationssuche oder Literaturrecherche.

- **Schnelle Antwortzeiten.** Schnelle Antwortzeiten haben bei interaktiv genutzten Systemen oberste Priorität. Ein Meta-Recherchesystem ist allerdings von den Antwortzeiten der zugrundeliegenden Dienste abhängig. Es kann die Ergebnisse nicht schneller liefern als die Basisdienste selbst. Da diese Dienste über die HTTP-Schnittstelle angefragt werden, können Antwortzeiten zudem recht lange dauern und sehr unzuverlässig sein.

2.4 Architektur für ein Meta-Recherchesystem

In diesem Abschnitt wird eine Architektur für ein Meta-Recherchesystem zur Literaturrecherche und -beschaffung vorgeschlagen. Diese Architektur wurde im Rahmen des UniCats-Projekts [CPSL99] entwickelt und trägt den im vorhergehenden Abschnitt erarbeiteten Anforderungen Rechnung. Abbildung 2.1 zeigt die wesentlichen Komponenten der Architektur, die zur Vermittlung zwischen den Benutzern und den zugrundeliegenden Recherchediensten vorgesehen sind. Die Architektur besteht aus drei Arten von Komponenten, die im Folgenden genauer beschrieben werden.

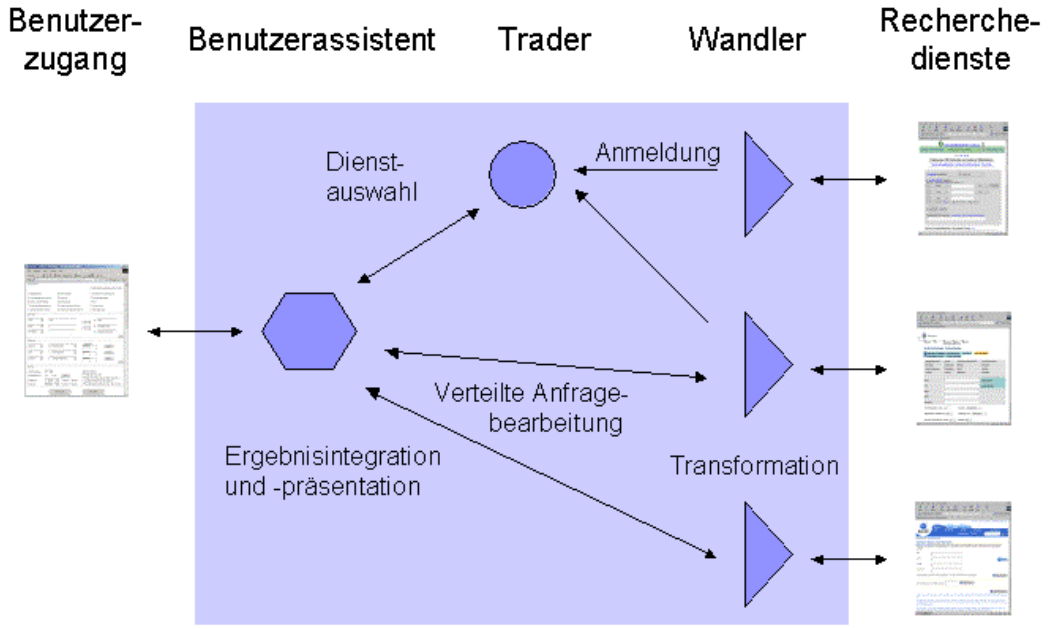


Abbildung 2.1: Überblick über die UniCats-Architektur

Wandler. Mit Hilfe von Wandlern [Pul01] werden die existierenden Recherchedienste in das Gesamtsystem eingebunden. Wandler bieten dem System intern eine einheitliche Anfrage- und Ergebnisschnittstelle an, über welche jeder Dienst in der gleichen Weise angesprochen werden kann. Die Wandler selbst kommunizieren mit den zugrundeliegenden Diensten über das HTTP-Protokoll, und zwar in der von den Diensten vorgegebenen proprietären Anfragesprache. Durch Wandler wird also die syntaktische und semantische Heterogenität der existierenden Recherchedienste überwunden.

Ein Wandler wird stets für einen bestimmten Recherchedienst erzeugt. Es ist möglich, für einen Dienst mehrere konkurrierende Wandler zu betreiben. Erst nachdem ein Wandler für einen Recherchedienst erzeugt wurde, ist dieser Dienst im Gesamtsystem bekannt und verfügbar. Nach der Generierung meldet der Wandler den Dienst bei einem Trader im System an und übermittelt diesem Trader die notwendigen Informationen, z.B. den Namen, die Zugangsadresse und einige wesentliche Dienstmerkmale.

Trader. Der Trader [Chr99] ist eine Art Gelber-Seiten-Dienst. Seine Aufgabe ist es, zu einer gegebenen Benutzeranfrage die geeigneten Dienste herauszufinden. Dazu verfügt er über Informationen zu allen im System verfügbaren Recherchediensten. Zu jedem Dienst speichert er diese Informationen in Form eines Profils ab. Ein solches Dienstprofil kann verschiedene Arten von Informationen enthalten [Chr01b], z.B. Angaben zum geführten Dokumentbestand, die Anzahl der verfügbaren Dokumente, die Art der Dokumente, die abgedeckten Fachbereiche oder auch die Sprachen der Dokumente. Weiterhin enthält ein Dienstprofil Informationen zu den Attributen eines Dienstes, d.h. welche Attribute anfragbar sind und welche zur Beschreibung eines Dokuments geliefert werden können. Auch technische und statistische Daten sind im Dienstprofil zu finden, beispielsweise die durchschnittlichen Antwortzeiten des Dienstes oder die durchschnittlichen Preise und Lieferzeiten der Dokumente. Diese Angaben können entweder manuell erstellt worden sein, z.B. vom Erzeuger des Wandlers, können aber auch automatisch vom Trader im laufenden Betrieb ermittelt werden.

Da im Gesamtsystem zahlreiche Dienste eingebunden werden können und die Menge der Informationen, die von einem Trader in einem Dienstprofil gespeichert wird, recht umfangreich werden kann, ist vorgesehen, dass es auch mehrere kooperierende Trader innerhalb des Systems geben kann [Chr01a], welche sich dann beispielsweise auf thematisch oder regional ähnliche Dienstgruppen spezialisieren können.

Benutzerassistent. Der Benutzerassistent [Sch02] stellt für den Benutzer den zentralen Zugangspunkt zum Gesamtsystem dar. Die Benutzungsoberfläche ist über das World-Wide Web zugänglich. Hier kann der Benutzer seine Anfrage formulieren und erhält anschließend die Gesamtergebnismenge in einer einheitlichen und integrierten Anzeigeform. Zur Ermittlung der Gesamtergebnismenge geht der Benutzerassistent folgendermaßen vor: Zunächst erfragt er beim Trader geeignete Recherchedienste für die Suchanfrage. Daraufhin

befragt er die Wandler der entsprechenden Dienste im einheitlichen internen Anfrageformat. Dies kann je nach Anfrage in paralleler Weise, nacheinander oder auch in einer kombinierten Form geschehen. Die einzelnen Ergebnismengen, die von den Wandlern zurückgeliefert werden, befinden sich bereits in einem einheitlichen Format. Nun erfolgt eine Integration auf Datensatzebene, d.h. zusammengehörige Teile von Datensätzen werden vereinigt und Duplikate werden erkannt. Die Gesamtergebnismenge wird schließlich dem Benutzer in einer angemessenen, graphisch aufbereiteten Darstellungsform präsentiert.

2.5 Fokus der Arbeit

2.5.1 Ziele der Arbeit

Die vorliegende Arbeit beschäftigt sich nicht mit allen Teilen der in Abbildung 2.1 gezeigten Architektur. Vielmehr greift sie das Problemfeld des Benutzerassistenten heraus, um für die Themenschwerpunkte Anfrageformulierung, Anfragebearbeitung sowie Ergebnisintegration und -präsentation geeignete Lösungen zu präsentieren.

Das Ziel der Arbeit besteht darin, dem Benutzer bei der wissenschaftlichen Literaturrecherche eine angemessene Form der Unterstützung im Umgang mit der vorhandenen Dienstevielfalt anbieten zu können. Dabei sollen auf der einen Seite möglichst viele der vorhandenen Informationen gesammelt und für den Benutzer entsprechend aufbereitet werden, so dass dieser bei seiner Entscheidungsfindung möglichst gut unterstützt werden kann. Gleichzeitig soll aber auf der anderen Seite auch die Performanz des Systems nicht unter der vollständigen Integration und Gewinnung der Synergien leiden, da sonst die Benutzerakzeptanz ebenfalls nicht gewährleistet werden kann.

Das angestrebte Optimierungsziel dieser Arbeit ist also vom Grundsatz her, dass alle gefundenen Dokumentreferenzen dem Benutzer sofort präsentiert werden, und zwar mit sämtlichen verfügbaren Informationen und komplett duplikatbereinigt. Auf Basis der heute existierenden Dienste und Technologien ist dies nicht möglich. Sämtliche momentan vorhandenen Meta-Recherchesysteme führen jeweils nur eine minimale Informationsintegration zugunsten schneller Antwortzeiten durch. Die vorliegende Arbeit wird untersuchen, inwieweit diese zweiteilige Forderung durch verschiedene Verfahren und Strategien der Anfragebearbeitung und der Duplikatbehandlung schon heute erfüllt werden kann. Auch umfassendere Konzeptionen in der Benutzerinteraktion sollen vorgenommen und dahingehend untersucht werden, ob sie geeignet sind, dem genannten Ziel näher zu kommen.

Danach sollen schließlich Prognosen und Empfehlungen gegeben werden, was beim Entwurf zukünftiger Einzeldienste und darauf aufsetzender Meta-Recherchesysteme beachtet werden kann, um die Informationssuche für den Benutzer zumindest systemseitig geeignet zu unterstützen und damit die Entwicklung zu „Informations-mündigen“ Bürgern in der Gesellschaft voranzutreiben.

2.5.2 Anforderungen an die Problemlösung

In Abschnitt 2.3 wurden die Anforderungen an ein Meta-Recherchesystem im Bereich der wissenschaftlichen Literatursuche herausgearbeitet. Sie wurden aus den Anforderungen der zugrundeliegenden Recherchedienste einerseits und den Anforderungen der Benutzer andererseits abgeleitet. In Abschnitt 2.4 wurde eine Architektur vorgestellt, die einen Rahmen bildet, der grundsätzlich die Erfüllung der genannten Anforderungen erlaubt. Da diese Arbeit nur ein Problemfeld aus diesem Rahmen herausgreift, nämlich den Bereich des Benutzerassistenten, wird die übrige Systemfunktionalität der Wandler und Trader als vorhanden vorausgesetzt.

Die aufgeführten Anforderungen können daher unterteilt werden in solche, die im Folgenden als erfüllt betrachtet werden sollen, und solche, die im weiteren Verlauf der Arbeit aufgegriffen und weiter bearbeitet werden.

Die Forderung der Dienstanbieter nach **Autonomie der Recherchedienste** erlaubt dem Meta-Recherchesystem **keinen Eingriff in die Recherchedienste**. Diese Anforderung betrifft vor allem die Wandler, die für die Anbindung der Recherchedienste an das Gesamtsystem zuständig sind. Dadurch, dass die Wandler auf den öffentlich, über das World-Wide Web zugänglichen Schnittstellen der Dienste aufsetzen, wird diese Voraussetzung erfüllt. Weiterhin sehen die im Rahmen des UniCats-Projekts entwickelten Wandler Mechanismen zum einfachen, semi-automatischen Erstellen der Wandler vor sowie zum Erkennen von Änderungen der HTML-Schnittstellen der Dienste [CSS02]. In vielen Fällen können sich die Wandler selbst durch entsprechende Maßnahmen an die neu formatierten oder strukturierten HTML-Seiten der Dienste anpassen, so dass sie bei einer **Evolution der Recherchedienste** nicht neu generiert werden müssen.

Um ein **umfassendes Dienstangebot** im Gesamtsystem integrieren zu können, d.h. um der geforderten **Offenheit** nachzukommen, dürfen nur minimale Anforderungen an die Recherchedienste gestellt werden. Dieser Forderung kommen in der vorgestellten Architektur ebenfalls die Wandler nach. Sie können nahezu beliebige Dienste, sofern diese über eine HTTP-Schnittstelle zugänglich sind, mit derselben einheitlichen Anfragefunktionalität und demselben einheitlichen Ergebnisformat versehen, so dass diese dem Gesamtsystem zur Verfügung stehen.

Die Kommunikation zwischen den einzelnen Komponenten der UniCats-Architektur erfolgt über einen Multicast-Mechanismus [CNSP00], über welchen XML-Nachrichten versendet werden. Diese Form der Kommunikation wurde speziell im Hinblick auf die **Dynamik** im Bereich der Literaturrecherche- und -beschaffungdienste gewählt und entsprechend weiterentwickelt und verfeinert. Dadurch können sich neue Dienste bzw. die dazugehörigen Wandler schnell im System bekannt machen und sich bei Bedarf auch wieder ebenso problemlos aus dem Gesamtsystem entfernen. Für die schnelle Erzeugung eines Wandlers wurde ein spezielles Verfahren entwickelt, welches als Generation-by-Example bezeichnet

werden kann, vgl. [Pul01] und [Sch01]. Ein Wandler wird also nicht durch mühsames Programmieren erstellt, sondern kann durch das Durchführen und Kommentieren einer Beispielrecherche bei einem Dienst automatisch erzeugt werden.

Die UniCats-Architektur sieht eine Trader-Komponente vor, deren Hauptaufgabe darin besteht, sämtliche verfügbaren Recherchedienste zu kennen und Informationen über diese zu sammeln, um zu einer Benutzeranfrage die besten Dienste empfehlen zu können. Trader gewährleisten damit die **Skalierbarkeit** des Systems, welche zusätzlich durch den eingesetzten Multicast-Mechanismus begünstigt wird. Falls ein einziger Trader die anfallende Informationsmenge nicht bewältigen kann, sind Kooperationen von Tradern möglich.

Die übrigen Forderungen fallen in den Kompetenzbereich des Benutzerassistenten, wobei bei einigen Anforderungen die Unterstützung der Trader und Wandler in Anspruch genommen werden kann.

Über eine Benutzungsschnittstelle im World-Wide Web soll der Benutzerassistent den Benutzern eine **einheitliche Anfragesprache und Ergebnisdarstellung** zur Interaktion mit dem Gesamtsystem anbieten. Die Anforderungen an die Anfragesprache bzw. den Mechanismus der Ergebnispräsentation gehen durchaus noch weiter. Beide sollen zum einen **einfach handhabbar** sein, zum anderen soll gleichzeitig aber auch die **Kontrollmöglichkeit** und damit auch ein gewisser Handlungsspielraum für den Benutzer gegeben sein. Der Anfragemechanismus muss beispielsweise sowohl für neue und ungeübte Benutzer als auch für regelmäßige Nutzer oder Experten geeignete Mechanismen vorsehen. Um dieses Spannungsfeld geeignet zu besetzen, sollen bekannte **menschliche Eigenschaften** (engl. human factors), d.h. physiologische, kognitive und emotionale Fähigkeiten, Fertigkeiten oder auch Grenzen des Benutzers herangezogen und berücksichtigt werden.

Die oben genannte **Einheitlichkeit** des Interaktionsmechanismus ist technisch zu verstehen. Dadurch wird die Heterogenität der zugrundeliegenden Recherchedienste vor dem Benutzer verborgen. Die dafür notwenige Aufgabe der **Transformation von Benutzeranfragen und Einzelergebnissen** teilt sich zwischen dem Benutzerassistenten und den verschiedenen Wandlern auf: Wandler stellen dem System intern eine einheitliche Anfragesprache für sämtliche Dienste zur Verfügung und liefern die Teilergebnisse in einem einheitlichen Format zurück. Demnach übernehmen sie einen Großteil der notwendigen Transformationsarbeit. Der Benutzerassistent muss eine möglicherweise komplexe Benutzeranfrage in eine Reihe von Anfragen im internen Format umsetzen. Ihm kommt somit mehr die **Koordinierung von Anfragen und Ergebnissen** zu. Dabei kann der Benutzerassistent zum Teil auf die Funktionalität des Traders zurückgreifen. Der Trader kann bei der Erstellung eines Anfrageplans über geeignete Recherchedienste Auskunft geben, sofern der Benutzer keine konkreten Dienste angegeben hat. Da es allerdings auch keine Garantie für das Vorhandensein oder die neutrale Handlungsweise des Traders geben muss, sollten Anfragepläne prinzipiell auch ohne die Inanspruchnahme eines Traders erstellt werden können.

Weiterhin muss der Benutzerassistent die empfangenen Teilergebnisse auf Datensatzebene integrieren. Dazu ist eine **Duplikatbehandlung** notwendig, d.h. zunächst müssen Duplikate erkannt werden und dann ggf. verschmolzen oder auf andere Art für den Benutzer kenntlich gemacht werden. Diese Integrationsarbeit ist rechentechnisch recht aufwendig und wird die Erstellung des Gesamtergebnisses sicherlich verzögern. Auch im Bereich der Anfragebearbeitung besteht sozusagen ein Spannungsfeld zwischen den Anforderungen. Auf der einen Seite sollen **schnelle Antwortzeiten** gewährleistet werden, auf der anderen Seite müssen die **Lastvorgaben** der Dienste berücksichtigt und auch exakt bei der Anfragebearbeitung eingehalten werden. Die Duplikatbehandlung ist von entscheidender Wichtigkeit, um Synergieeffekte nutzbar zu machen und die Entscheidungsfindung des Benutzers zu unterstützen, trägt aber sicherlich nicht zur Beschleunigung des Antwortzeitverhaltens bei.

Einzeln betrachtet, sind die Anforderungen an den Benutzerassistenten mit heute bekannten Verfahren und Techniken erfüllbar. Die Schwierigkeit bzw. die Herausforderung der vorliegenden Arbeit besteht allerdings darin, den gesamten Anforderungskatalog möglichst gut zu erfüllen. Die Akzeptanz und der Erfolg eines Meta-Recherchesystems hängt nämlich von allen genannten Anforderungen ab. Der Anforderungskatalog benennt sozusagen die kritischen Erfolgsfaktoren eines Meta-Recherchesystems. Ziel ist also die „gesamt-optimale“ Erfüllung des spezifizierten Anforderungskatalogs. Dieses Optimierungsproblem ist z.B. mit dem allgemeinen IR-Gütemaß vergleichbar, wo zur Optimierung gleichzeitig der Precision- und der Recall-Wert erhöht werden müssen.

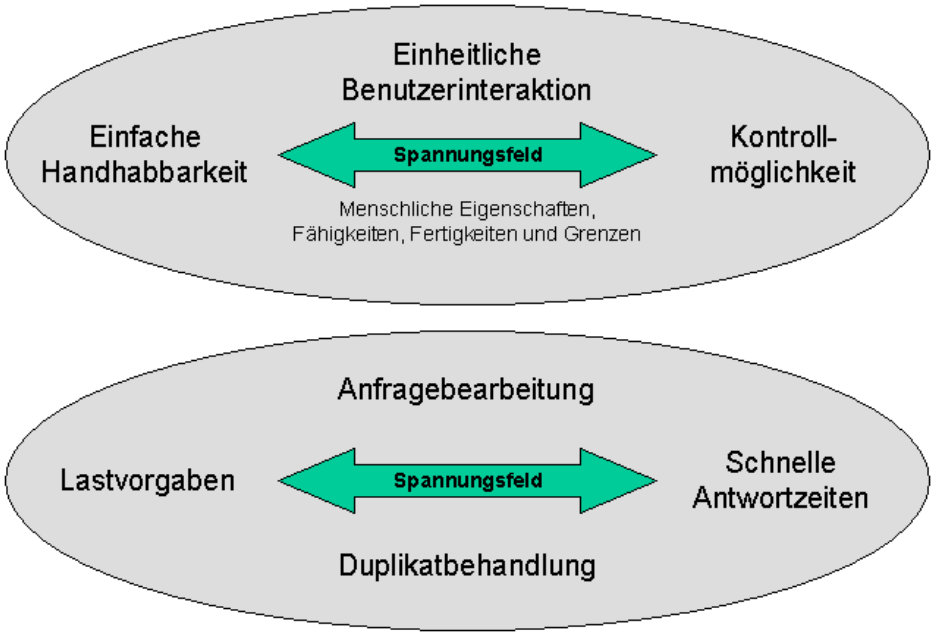


Abbildung 2.2: Spannungsfelder innerhalb des Anforderungskatalogs

Abbildung 2.2 verdeutlicht noch einmal die zwei Spannungsfelder, die sich im genannten Anforderungskatalog bemerkbar machen. Diese Problembereiche sind nur mit neuen tragfähigen Konzepten und systematischen Vorgehensweisen zu bewältigen. Die Untersuchung und Analyse, inwieweit sich nun ein gesamt-optimaler Lösungsansatz entwickeln lässt, soll Gegenstand der vorliegenden Arbeit sein. Gerade im Bereich der Anfragebearbeitung sollen dazu quantitative Aussagen und Erkenntnisse gewonnen werden, wohingegen die geeignete Gestaltung der Benutzungsoberfläche eher konzeptionelles Geschick und die Einbeziehung zahlreicher Benutzerstudien erfordern wird.

2.6 Resümee

In diesem Kapitel wurde anhand eines Szenarios in das Umfeld, die aktuelle Situation und die Problematik der wissenschaftlichen Literaturrecherche eingeführt. Dabei wurden zunächst die verschiedenen Arten von existierenden Recherche- und -beschaffungsdiensten vorgestellt. Anschließend wurden charakteristische Benutzeranfragen während einer wissenschaftlichen Literaturrecherche anhand von einigen Beispielen illustriert. Bei diesen Ausführungen wurde deutlich, dass in diesem Bereich ein Meta-Recherchesystem eine erhebliche Verbesserung der Situation bedeuten und für einen Benutzer einen unverzichtbaren Mehrwert liefern würde.

Danach wurde eine ausführliche Analyse der Anforderungen an ein derartiges Meta-Recherchesystem angestellt. Eine Liste der Anforderungen konnte über die zwei beteiligten "Parteien", Benutzer und zugrundeliegende Recherchedienste, hergeleitet werden. Daraufhin wurde im nächsten Schritt eine Gesamtarchitektur vorgestellt, die dieser Arbeit zugrundegelegt wird. Sie stellt einen Rahmen für die Erfüllung der erarbeiteten Anforderungen dar. Die einzelnen Komponenten der Architektur wurden ausführlich erläutert.

Der nächste Abschnitt machte deutlich, mit welchem Bereich der Architektur sich die vorliegende Arbeit beschäftigt. Dazu wurde noch einmal formuliert, welche der genannten Anforderungen damit als erfüllt betrachtet und welche im weiteren Verlauf der Arbeit noch detaillierter aufgegriffen werden.

Dabei wurde deutlich, wie unterschiedlich und teilweise auch widersprüchlich die beteiligten Interessen und damit auch Anforderungen sind: Die Anbieter von Recherchediensten wollen einen direkten Vergleich mit ihren Konkurrenten möglichst vermeiden. Sie sind zwar froh, dass ihnen die Teilnahme an einem derartigen System mehr Benutzer und eine größere Bekanntheit einbringt. Allerdings sind sie dafür auf keinen Fall bereit, Änderungen an ihren Diensten vorzunehmen und tolerieren die dadurch entstehende Mehrauslastung auch nur in einem gewissen, von ihnen vorgegebenen Rahmen. Die Benutzer haben dagegen ein starkes Interesse an der direkten Vergleichbarkeit der Angebote. Daher wollen sie möglichst viele relevante Informationen zur Entscheidungsfindung heranziehen, aber natürlich nicht von einer Informationsflut überschüttet werden und schon gar nicht lange warten. In dieser Gratwanderung zwischen eigentlich schon in sich widersprüchlichen Benutzerwünschen und

den Gegebenheiten und Vorgaben der Dienstanbieter liegt die Herausforderung und der Gestaltungsspielraum der vorliegenden Arbeit.

3 Stand der Technik

3.1 Übersicht

Der im vorangegangenen Kapitel erarbeitete Anforderungskatalog gruppiert sich im Wesentlichen um zwei Schwerpunkte: Gefordert ist eine geeignete Benutzerinteraktion, welche einfache, intuitive, aber auch ausdrucksstarke Mechanismen für die Anfrageformulierung des Benutzers vorsieht. Die Kontrollmöglichkeit und Ausdrucksstärke sollen den Benutzer vor allem vor einer Informationsüberflutung schützen. Bei der Konzeption der Benutzerinteraktion sind menschliche Fähigkeiten und Grenzen zu berücksichtigen, wobei diese Forderung durch Ergebnisse aus Benutzerstudien noch weiter konkretisiert werden soll. Der zweite Schwerpunkt des Anforderungskatalogs betrifft die Anfragebearbeitung. Hier sollen die Lastvorgaben der Einzeldienste berücksichtigt, aber dennoch schnelle Antwortzeiten hervorgebracht werden. Neben der vollständigen Anfrageplanung und -bearbeitung ist eine Duplikatbehandlung vorzusehen, welche die Qualität des Gesamtergebnisses erhöht, aber möglichst wenig Einfluss auf die Antwortzeiten des Meta-Recherchesystems haben sollte. Damit ergibt sich folgender Anforderungskatalog:

- A1: Einheitliche, integrierte Benutzungsschnittstelle
- A2: Einfache, intuitive Benutzerinteraktion
- A3: Ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit
- A4: Berücksichtigung menschlicher Eigenschaften (im Sinne von Fähigkeiten, Fertigkeiten und Grenzen)
- A5: Vollständige Anfrageplanung und -bearbeitung
- A6: Duplikatbehandlung
- A7: Schnelle Antwortzeiten
- A8: Berücksichtigung von Lastvorgaben

Der Anforderungskatalog soll im Verlauf dieses Kapitels zur Betrachtung und Bewertung existierender Systeme und Ansätze herangezogen werden. Dadurch soll identifiziert werden, an welchen Stellen bereits gute Lösungen existieren und wo der größte Handlungsbedarf für die vorliegende Arbeit besteht. Dazu werden zunächst Benutzerstudien betrachtet (Abschnitt 3.2), um einen Eindruck von den zu erwartenden menschlichen Eigenschaften zu gewinnen, die bei den technischen Konzeptionen nicht vernachlässigt werden dürfen. In diesem Abschnitt soll die Forderung nach der Berücksichtigung menschlicher Fähigkeiten, Fertigkeiten und Grenzen genauer spezifiziert werden.

Im weiteren Verlauf des Kapitels werden dann existierende Ansätze und Techniken betrachtet, die den aufgestellten Anforderungskatalog – zumindest in Teilen – erfüllen. Zunächst beschäftigen wir uns mit existierenden Meta-Recherchesystemen in der Praxis (Abschnitt 3.3) und prüfen, inwieweit sie die genannten Anforderungen erfüllen bzw. an welchen Stellen der größte Handlungsbedarf besteht. Anschließend betrachten wir Ansätze aus dem Bereich der I^3-Architekturen (Abschnitt 3.4), welche technische Lösungen für die Intelligente Integration von Informationsquellen bereitstellen. Danach wenden wir uns speziellen Konzepten zur Benutzerunterstützung zu (Abschnitt 3.5) und betrachten zum einen Personalisierungstechniken, die zur Unterstützung der Anfrageformulierung und zur Verhinderung einer Informationsüberflutung eingesetzt werden, und zum anderen graphische und gestalterische Konzepte, die zur anschaulichen Aufbereitung und besseren Handhabbarkeit von größeren Ergebnismengen verwendet werden.

3.2 Benutzerstudien

In diesem Abschnitt sollen Ergebnisse von Benutzerstudien herangezogen werden, um die zu berücksichtigenden menschlichen Eigenschaften (A4) bei einer Informationssuche weiter zu konkretisieren. In diesem Zusammenhang wollen wir zwei Arten von Benutzerstudien betrachten: zum einen Studien, die versuchen, die Informationssuche als Prozess zu charakterisieren. Diese Sichtweise ist mittlerweile weit verbreitet und auch hinreichend wissenschaftlich durch Studien untermauert worden. Und zum anderen sind Studien interessant, die das tatsächliche Rechercheverhalten der Benutzer beobachten und auswerten, sogenannte Transaction Log Analysen. Sie verzichten auf Beobachtungen und Interpretationen und konzentrieren sich auf das real erfassbare Verhalten der Benutzer. Sie spiegeln die aktuelle Situation im Bereich der Informationssuche am besten wider.

3.2.1 Der Prozesscharakter der Informationssuche

Die Studien, die hier vorgestellt werden, beziehen sich auf die Informationssuche eines Studenten bzw. Wissenschaftlers bei der Anfertigung einer wissenschaftlichen Arbeit. Allgemeine Studien über das Verhalten der Benutzer im WWW werden hier nicht berücksichtigt, da die Erwartungen bei dieser Form der allgemeinen Suche zu unterschiedlich sein können.

Kuhlthau hat ein Prozessmodell der Informationssuche entwickelt, welches sich am kognivistischen Lernprozess orientiert. In ihrem Buch [Kuh93] hat Carol Kuhlthau diesen Suchprozess genauer beschrieben und ihre Theorie durch mehrere Benutzerstudien belegt. Dem von außen zu beobachtenden Informationssuchprozess entspricht im Inneren des Benutzers ein kognitivistischer Lernprozess.

Bei diesem Prozess lassen sich sechs charakteristische Phasen identifizieren:

- Vorbereitung (*task initiation*)
- Themenauswahl (*topic selection*)
- Orientierung (*pre-focus exploration*)
- Schwerpunktformulierung (*focus formulation*)
- Sammeln von Informationen (*information collection*)
- Abschluss (*search closure*)

Die einzelnen Phasen werden in der Regel nicht linear, sondern oft auch wiederholt durchlaufen. Dabei kann keine Phase wirklich übersprungen werden. Generell bedeutet der Eintritt in eine höhere Phase eine abnehmende Unsicherheit auf Seiten des Benutzers. Die Unsicherheit und das Ungewisse, zu welchem Ergebnis eine Informationssuche führen wird, sind von zentraler Bedeutung im Suchprozess.

Themenauswahl und Schwerpunktformulierung sind besonders hervorzuheben, da in diesen Phasen wichtige Entscheidungen getroffen werden müssen. Dabei wurde herausgefunden, dass Studenten ihre Entscheidungen aufgrund von vier Kriterien treffen: persönlichem Interesse, vorgegebenen Anforderungen, verfügbaren Informationen und aufgewendeter Zeit. Diese Kriterien wurden beim Nachfragen von den Studenten explizit genannt.

- Persönliches Interesse: Dieses steigt in der Regel, wenn die Suche gut bzw. erfolgreich verläuft.
- Vorgegebene Anforderungen: Die Anforderungen oder das Ziel der Suche wird meist von einem Lehrer oder Betreuer vorgegeben.
- Verfügbare Informationen: Der Student verwendet in der Regel nur die Informationen, die in den naheliegenden Katalogen und Suchdiensten verfügbar sind.
- Aufgewendete Zeit: Die Zeit, die eine Informationssuche in Anspruch nimmt, wird meistens unterschätzt. Dieser Faktor wird im Verlauf der Suche immer wichtiger, negative Empfindungen kommen oft in diesem Zusammenhang zum Ausdruck.

Während der Informationssuche wurden zwei vorherrschende Grundstimmungen erkannt: Die Grundstimmung „einladend“ (*invitational*) begünstigt eine offene Suche, der Benutzer ist offen für neue Informationen. Die Grundstimmung „hinweisend“ (*indicative*) begünstigt den Suchabschluss, der Benutzer will keine neuen Hinweise, Anregungen oder Informationen mehr finden.

Marchionini [Mar95] entwickelte ein umfassendes Modell der Informationssuche in elektronischen und gedruckten Medien. Das Modell ist in seiner Allgemeinheit ein sehr geeignetes Rahmenmodell der Informationssuche. Der Prozess gliedert sich in acht Teilprozesse:

- Erkennen und Akzeptieren eines Informationsproblems (*recognize, accept*)
- Verstehen und Eingrenzen des Problems (*define problem*)
- Auswählen eines Suchsystems (*select source*)
- Formulieren einer Suchanfrage (*formulate query*)
- Ausführen der Suche (*execute query*)
- Lesen und Bewerten der Ergebnisse (*examine results*)
- Extrahieren der Information (*extract information*)
- Reflektieren, Iterieren, Beenden (*reflect, stop*)

Folgende Abbildung 3.1 zeigt die Verknüpfung der einzelnen Prozessphasen bzw. -schritte (vgl. [Mar95], S. 50). Dabei werden die einzelnen Phasen meist wiederholt durchlaufen, genauso wie im Prozessmodell von Carol Kuhlthau. Marchionini kann allerdings zusätzlich zumindest grobe Angaben zu den Wahrscheinlichkeiten bestimmter Phasenübergänge machen. Die durchgezogenen Pfeile beschreiben in der folgenden Abbildung die wahrscheinlicheren Phasenübergänge.

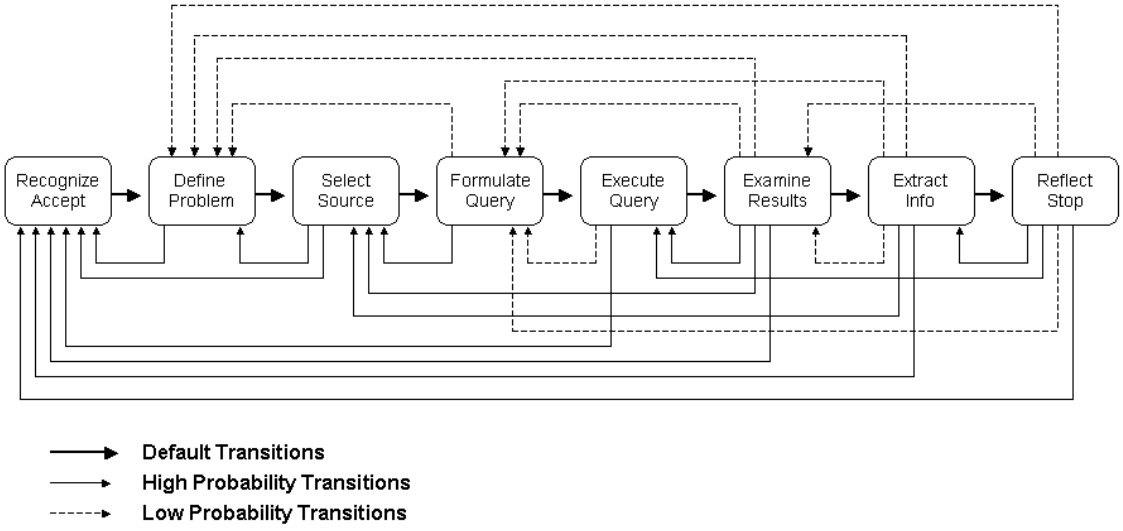


Abbildung 3.1: Der Informationssuchprozess nach Marchionini

Das Modell sieht ebenso wie Kuhlthaus Modell kein vorwärtsgerichtetes Überspringen von Teilprozessen vor. Insofern kann der Prozess nicht beendet werden, wenn keine nützlichen Informationen herausgezogen werden konnten.

Marchionini betont, dass das Vorgehen bei einer Informationssuche – trotz des allgemeinen Prozessmodells – doch individuell höchst verschieden sein kann. Dabei spielen die kognitiven Anlagen eines Menschen eine wesentliche Rolle (*general cognitive facility*), d.h. die

Intelligenz, die Vorlieben und die Handlungsmuster, die beispielsweise eher strategisch oder eher intuitiv geprägt sein können. Weiterhin ist ausschlaggebend, über welches Wissen und welche Fähigkeiten der betreffende Benutzer im Anwendungsbereich besitzt (*domain expertise*), im Umgang mit dem Recherchesystem (*system expertise*) und mit dem Vorgang der Informationssuche (*information seeking expertise*).

Ellis [EH97] hat das Informationssuchverhalten anhand von Fragebögen untersucht und ein Modell entwickelt, welches anhand von verschiedenen Berufssparten validiert wurde. Dazu untersuchte Ellis mehrere Gruppen von akademischen Wissenschaftlern sowie eine Gruppe von Ingenieuren und Wissenschaftlern in der Industrie. Ellis formuliert und beschreibt das gefundene Modell folgendermaßen:

- *Surveying:* In dieser Phase verschafft man sich einen Überblick über ein neues Gebiet. Die häufigsten Mittel sind dabei eine retrospektive Literaturrecherche oder das Aktivieren von persönlichen Kontakten, bei denen man Expertise auf diesem Gebiet vermutet. Durch die Literaturrecherche will man erfahren, was in den letzten Jahren auf dem Gebiet getan wurde, was der aktuelle Stand ist und wo das eigene Projekt neue Impulse geben könnte. Die kontaktierten Personen können Kollegen oder andere Bekannte sein (*formal and informal information channels*).
- *Chaining:* In dieser Phase werden ausgehend von bekannten Diensten und Papieren weitere aufgespürt. Bei Papieren werden dazu gerne die Referenzen oder Autoren verwendet. Von bekannten Personen kann sich der Informationssuchende auch weitere Experten empfehlen lassen. Die Phase wird entweder durch die zur Verfügung stehende Zeit beendet oder durch die Menge der Kontakte, die gerade verfügbar ist.
- *Monitoring:* Hierbei ist das Ziel, auf dem Laufenden zu bleiben. Das kann durch die Nutzung von Benachrichtigungsdiensten geschehen, das Abonnement von Zeitschriften, den Besuch von Konferenzen und den Austausch mit anderen Personen auf dem Gebiet. Gerade das Gespräch mit anderen Personen wird als sehr nützlich weil zeitsparend bewertet im Gegensatz zu einer Literaturrecherche.
- *Browsing:* Das Stöbern ist sehr wichtig. Der Informationssuchende schaut sich Umläufe an, Zeitschriftenhefte, Inhaltsverzeichnisse, Listen mit Referenzen. Dieses Stöbern kann einerseits sehr spontan und zufällig geschehen, das Verfolgen von Referenzen ist dagegen eher zielgerichtet.
- *Differentiating:* In dieser Phase wird die Information nach ihrer Relevanz zum aktuellen Projekt bewertet. Hier spielt der Zeitfaktor und die Informationsfülle eine große Rolle. In der Regel werden Diskussionen mehr geschätzt als Literaturstudien.
- *Filtering:* Hier werden Kriterien gesucht, mit denen man die Informationssuche so relevant und präzise wie möglich machen kann. Dabei besteht der Konflikt zwischen der zur Verfügung stehenden Zeit und dem Risiko, wichtige Informationen zu

übersehen. Stichworte sind das häufigste Kriterium, die Zeitdauer ist als zweites zu nennen.

- *Extracting:* In dieser Phase werden die verschiedenen Quellen zusammengesucht, je nach Aufgabe bzw. je nach zu erstellendem Dokument: Abschlussbericht, Technischer Bericht oder Vortrag.
- *Ending:* Das ist die Phase kurz bevor das Projekt beendet wird. Hier werden noch einmal die neuesten Entwicklungen berücksichtigt und offene Fragen bearbeitet. Der Fokus ist sehr bestimmt und die Recherchen finden nur noch in kleinem Maße statt.

Die Untersuchungen haben ergeben, dass sowohl formale als auch informale Informationskanäle verwendet werden. In der ersten Phase wird von beiden Möglichkeiten reger Gebrauch gemacht. Von Phase zu Phase werden die Informationssuchenden immer spezieller und damit selektiver. Dabei scheint der formale Anteil abzunehmen und der informelle Anteil immer wichtiger zu werden. In der abschließenden Phase werden dann noch einmal beide Aspekte benötigt, wenn auch in kleinerem Maße. Insgesamt ist auch dieser Prozess iterativ zu sehen und gerade bei länger angelegten Projekten sind mehrere Durchläufe und mehrere kleinere Zyklen zu erwarten.

Bei Untersuchungen von Wissenschaftlern und Ingenieuren wurden einige Unterschiede deutlich: Ingenieure betreiben mehr interne Kommunikation, fragen zuerst die eigenen Kollegen oder die beteiligten Kunden und Verkäufer. Sie nutzen weniger die allgemein verfügbaren Informationsdienste, da sie viel mit internen Berichten und Dokumenten zu tun haben. Wissenschaftler dagegen beschaffen sich ihre Informationen mehr außerhalb ihres Umfelds, im ersten Schritt über eine Literaturrecherche und dann über auswärtige Kollegen und Kontakte. Das auf dem Laufenden Bleiben ist im wissenschaftlichen Bereich von größerer Bedeutung. Das aufgestellte Modell, was die beobachtbaren Verhaltensmuster in jeder Prozessphase beschreibt, passt allerdings auf beide Gruppen. In jeder Phase kann man dieselben Vorgehensweisen und Überlegungen beobachten, nur sind sie prozentual unterschiedlich auf die Gruppen verteilt.

Bewertung

Die genannten Ansätze liefern interessante und wissenschaftlich fundierte Ergebnisse und helfen beim Verstehen der Art und Komplexität der Vorgänge, die sich bei einer Informationssuche auf Seiten des Benutzers abspielen. Alle vorgestellten Ansätze modellieren die Informationssuche als langandauernden und iterativen Prozess, wenn sich auch die Anzahl und Bezeichnung der identifizierten Phasen geringfügig unterscheiden. Allerdings legen die Ansätze jeweils spezielle Schwerpunkte bei ihren Betrachtungen: So findet laut Kuhlthau während der Informationssuche beim Benutzer ein kognitivistischer Lernprozess statt, in welchem sich die Unsicherheit des Benutzers phasenweise reduziert. Kuhlthau identifiziert außerdem noch die vier wichtigsten Entscheidungskriterien, die erstaunlicherweise recht

wenig mit der inhaltlichen Ausrichtung und Eignung der Dokumente zu tun haben. Marchionini betont, dass das Durchführen einer Informationssuche oft stärker von den individuellen Eigenschaften eines Benutzers geprägt ist als von den identifizierten allgemeinen Prozessphasen. Und Ellis rückt in den Vordergrund, dass in jeder Prozessphase formale und informale Informationskanäle benutzt werden und beide gleichsam von entscheidender Bedeutung sind.

Ein Ansatzpunkt zur Unterstützung des Benutzers ist sicherlich dieser Prozesscharakter der Informationssuche, den es dementsprechend von der technischen Seite zu berücksichtigen gilt. Eine Möglichkeit wäre hier, den Benutzer durch technische Hilfsmittel durch den Prozess zu begleiten und an geeigneten Stellen auch mit speziellen Maßnahmen und Werkzeugen zu unterstützen. Eine Schwierigkeit wird allerdings auch sofort klar: Zum einen ist es schwierig, zu entscheiden, welches Prozessmodell zugrundegelegt werden soll. Da die Prozesse nun höchst individuell und flexibel durchlaufen werden, ist es zum anderen nahezu unmöglich, die nächste Aktion des Benutzers vorhersagen zu wollen. Möglicherweise ist auch ein einziges Recherchesystem oder gar ein Meta-Recherchesystem, wie es in dieser Arbeit angestrebt wird, nicht in der Lage, den kompletten Prozess zu begleiten und alle benötigten Recherchequellen integriert zur Verfügung zu stellen. Die informalen Informationskanäle können allerdings gewiss nicht von einem System zur Verfügung gestellt werden. Insofern wird auch klar, in welchem Maße und in welchem Bereich eine technische Unterstützung des Prozesses nur Einfluss nehmen kann und wo die Grenzen einer technischen Unterstützung zu erwarten sind.

Im Gegenzug dazu könnte man aber auch am Verhalten des Benutzers ansetzen und versuchen, dieses möglichst individuell zu unterstützen. Dabei ist es wichtig, persönliche und situative Eigenheiten, Vorlieben und Fähigkeiten des Benutzers zu kennen, vielleicht sogar dessen persönliche Entscheidungskriterien. In diesem Fall könnten die identifizierten Phasen des Informationsprozesses dafür dienen, die Menge und Ausrichtung der zu erwartenden Anfragen zu ermitteln, die prinzipiell von einem Recherchesystem unterstützt werden sollen: beispielsweise breite Überblicksrecherchen genauso wie sehr spezielle Informationswünsche, die ggf. noch verfeinert werden sollen, oder auch schnelle gezielte Auskünfte zu bestimmten Dokumenten.

Schließlich sollte man dem Prozesscharakter in jedem Fall jedoch insofern Rechnung tragen, dass Zwischenergebnisse oder Anfragen gespeichert werden können, um dem Benutzer ein späteres Aufsetzen zu ermöglichen.

3.2.2 Transaction Log Analyse

Eine weitere Herangehensweise zur Charakterisierung der Informationssuche besteht darin, an existierenden Recherchesystemen die Interaktionen der Benutzer zu beobachten. Eine weit verbreitete Form, Untersuchungen über das Verhalten der Benutzer durchzuführen, ist die *Transaction Log Analyse* (TLA). Dabei werden sämtliche von den Benutzern getätigten

Interaktionen mit dem System aufgezeichnet und anschließend ausgewertet. Aufgezeichnet werden dabei die gestellten Anfragen mit all ihren Optionen und die betrachteten Ergebnisse. Die Benutzer werden dabei in der Regel weder mit der Kamera beobachtet noch werden sie hinterher befragt.

Der große Vorteil dieser Studien liegt darin, dass die Systeme und die Benutzer im real laufenden Betrieb beobachtet werden, also in keiner Testumgebung und nicht bei der Durchführung von Testaufgaben. Bei dieser Vorgehensweise werden keine Vermutungen über die Ziele und Absichten der Benutzer angestellt, sondern lediglich die tatsächlichen Aktionen in Betracht gezogen.

Mittlerweile ist es technisch möglich und auch weit verbreitet, die Benutzerinteraktionen mitzuprotokollieren. Auf lokalen Rechnern ist dies einfach und schon längere Zeit möglich. Gerade bei Diensten, auf die die Benutzer sozusagen anonym über das Internet bzw. das WWW zugreifen, ist dies schon schwieriger: Das Problem besteht darin, die einzelnen Benutzeraktionen zu Sitzungen desselben Benutzers zusammenzufassen. Dazu ist es nicht ausreichend, nur die IP-Adresse des zugreifenden Rechners zu berücksichtigen. Gerade bei großen Internetanbietern wie z.B. T-Online verbergen sich viele Benutzer hinter derselben IP-Adresse. Verbessern kann man die Erkennungsleistung dadurch, dass noch der Browsertyp hinzugenommen wird. Um sicher sein zu können, sollte man allerdings Cookies auf dem Rechner des Benutzers setzen, auch wenn das nicht alle Benutzer zulassen. Schließlich muss man noch eine Zeitschranke festlegen, ab der eine Sitzung als unterbrochen bzw. beendet angesehen werden soll.

Natürlich ist es bei dieser Vorgehensweise nicht möglich, allgemeingültige Aussagen über „den“ Benutzer beim Suchen zu treffen. Anhaltspunkte können allerdings gefunden werden, wenn man konkrete Benutzergruppen an konkreten Benutzungsschnittstellen mit bestimmten Dokumentsammlungen interagieren lässt. So sind auch viele der Untersuchungen angelegt. Diese punktuellen Studien können einen guten Eindruck vom Benutzerverhalten vermitteln. Wenn in verschiedenen Studien ähnliche Ergebnisse gefunden werden, untermauert das natürlich die gefundenen Aussagen und macht sie dadurch schon weitaus allgemeingültiger.

Im Folgenden werden daher Studien aus zwei Bereichen vorgestellt: Zuerst über den Umgang der Benutzer mit den üblichen Suchmaschinen im World-Wide Web. Hierbei handelt es sich um die allgemeinste Form der Informationssuche. Dann fokussieren wir auf den spezielleren Bereich der Literaturrecherche. Dazu werden einige aktuelle Studien zur Nutzung von Digitalen Bibliotheken im Bereich der Informatikliteratur betrachtet.

3.2.2.1 Benutzerstudien von Suchmaschinen im WWW

Die Suche im WWW ist zu einem wichtigen Thema geworden, das nicht nur Informatiker betrifft. So haben auch ungeübte Benutzer seit 1993 Fähigkeiten erworben bzw. erwerben müssen, die früher beispielsweise eher Bibliothekaren vorbehalten waren. Eine Studie hat

ergeben [Sul01], dass Amerikaner etwa jeden zweiten Tag (13 Mal im Monat) eine Suche im Internet durchführen, ein Drittel der Befragten macht sogar eine oder mehrere Suchen pro Tag.

Die Dokumentsammlung im WWW kann man als sehr vielfältig bezeichnen, die Benutzergruppe ist recht groß und sehr unterschiedlich, wobei sicherlich bestimmte Gruppen (z.B. Jugendliche, Informatiker, ...) überrepräsentiert sind. Die Dokumente werden im Volltext indiziert, vergleichbar also mit IR-Systemen, allerdings ist das WWW thematisch breiter gestreut. Zur Zeit sind drei Studien über das Suchverhalten im WWW veröffentlicht worden. Im Folgenden werden die Ergebnisse kurz zusammengestellt.

Die Excite-Studie [JSS00]: Die Excite-Suchmaschine (http://www.excite.com) entstand im Jahre 1994. Sie zählte bis Dezember 2001 zu den größten Suchmaschinen weltweit und indizierte beispielsweise im September 2001 rund 380 Millionen Webseiten. Seit Dezember 2001 werden die Suchergebnisse von Excite durch die bezahlten Eintragungen vom Overture-Dienst dominiert. Für die vorliegende Studie wurden die Recherchedaten von einem Tag gesammelt. Die Daten stammen vom 9. März 1997, damals betrug der indizierte Datenbestand in etwa 40 Millionen Webseiten.

In diesem Zeitraum wurden 51 473 Anfragen von 18 113 Benutzern gestellt. Die Anfragen wurden als *eindeutig* (35%), *modifiziert* (22%) und *identisch* (43%) klassifiziert. Als eindeutig wird die erste Anfrage eines Benutzers bezeichnet, als modifiziert werden alle nachfolgenden Anfragen desselben Benutzers klassifiziert. Derartige Anfragen entstehen in der Regel durch Hinzufügen, Weglassen oder Ändern von Anfragetermen. Die Anzahl der eindeutigen Anfragen entspricht der Anzahl der Benutzer. Der hohe Anteil identischer Anfragen lässt sich folgendermaßen erklären: Natürlich ist es möglich, dass Benutzer dieselbe Frage ohne Änderungen nochmals eingeben. Das wurde auch in anderen Studien nachgewiesen. In der vorliegenden Studie kann man daraus vielmehr schließen, dass der Benutzer eine weitere Ergebnisseite mit Treffern angefordert hat. Beim Durchlaufen der Ergebnisseiten generiert Excite intern dieselbe Anfrage, nur wird ein anderer gewünschter Trefferbereich angefordert.

Im Rahmen einer Sitzung wurden durchschnittlich 2,84 Anfragen gestellt. Ignoriert man die identischen Anfragen, die ja genau genommen der Ergebnisbegutachtung zuzurechnen sind, ist die durchschnittliche Sitzung 1,6 Anfragen lang. Allerdings werden in der Studie keine Angaben darüber gemacht, wie eine Sitzung definiert wurde. Weder wird eine Zeitangabe genannt noch ist eine Aussage getroffen worden, ob zur Identifizierung eines Benutzers auf Basis von IP-Adressen oder mit Hilfe von Cookies gearbeitet wurde. Da sich der Zeitpunkt der Erhebung lediglich auf einen Tag beschränkt, liegt die Vermutung nahe, dass sämtliche Anfragen einer IP-Adresse zu einer Sitzung zusammengefasst wurden.

Im Allgemeinen wurden die Anfragen nicht häufig modifiziert, 67 % der Benutzer stellten nur eine Anfrage. Lediglich 14 % der Benutzer stellten drei oder mehr Anfragen. Wenn es zu

Modifikationen kam, wurden am häufigsten Terme verändert (35%), wobei die Anzahl der Terme insgesamt gleich blieb. In 19 % der Modifikationen wurde ein Term hinzugefügt, in 16 % wurde ein Term weggelassen.

Aufgrund der identischen Anfragen kann man Erkenntnisse darüber erhalten, wie sich der Benutzer beim Anschauen der Ergebnisse verhält. Bei Excite werden immer zehn Treffer auf einer Ergebnisseite präsentiert. Im Durchschnitt wurden 2,35 Seiten angeschaut, 58 % der Benutzer schauten nur die erste Ergebnisseite an. An dieser Stelle kann eine TLA leider keine Erklärungen liefern. Möglicherweise waren die Ergebnisse so gut, dass unter den ersten Treffern das Gesuchte enthalten war oder aber die Ergebnisse lagen so neben den Erwartungen, dass der Benutzer die Suchaktion sofort beendet hat. Oder der Benutzer hat die Anfrage umformuliert.

Nun zu den Anfragen selbst, die gestellt wurden: sie enthielten im Durchschnitt 2,21 Terme. 62 % der Anfragen enthielten ein oder zwei Terme. Die Zahlen sind sogar recht hoch, da die Operatoren dabei noch mit eingerechnet wurden. Im Allgemeinen wurden boolesche Operatoren selten benutzt (18%), der AND Operator war unter ihnen der häufigste (14%). In etlichen Fällen wurden die Operatoren fehlerhaft angewendet.

In Tabelle 3.1 werden die Ergebnisse der Excite-Studie nochmals zusammengefasst und den im Folgenden vorgestellten Studien gegenüber gestellt.

Die Altavista-Studie [SHMM99]: Die Altavista-Suchmaschine (http://www.altavista.com) ist eine der ersten und bekanntesten Suchmaschinen. Noch heute gehört sie zu den größten fünf Suchmaschinen weltweit. Altavista indizierte im Dezember 2001 etwa 550 Millionen Webseiten.

Diese Studie ist im Hinblick auf den Erhebungszeitraum, die Menge der Anfragen und die Qualität der Auswertungen die wohl umfassendste, die bisher in diesem Bereich durchgeführt wurde. Sie wurde über einen Zeitraum von 6 Wochen durchgeführt (vom 2. August bis 13. September 1998) und beinhaltet 1 Milliarde Anfragen. Zum damaligen Zeitpunkt wurden von Altavista ca. 140 Millionen Webseiten indiziert.

Im Zeitraum der Studie wurden 285 Millionen Recherchesitzungen identifiziert. Eine Sitzung soll hier die Vorgehensweise beschreiben, die ein Benutzer unternimmt, um sein Informationsbedürfnis zu einem bestimmten Thema zu befriedigen. Die Identifizierung der Sitzungen erfolgt in dieser Studie sowohl über Cookies als auch über Zeitschranken. 96 % der Benutzer haben Cookies zugelassen, die Auswertungen der Ergebnisse mit und ohne Cookies unterschieden sich nicht markant und wurden daher in der Studie gemeinsam aufgeführt. Eine Sitzung ist vom selben Benutzer, wenn nicht mehr als 5 Minuten Pause zwischen zwei Anfragen liegen. Durch diese Heuristik wird eine Sitzung tendenziell als zu lang angenommen, da in einer Sitzung durchaus auch mehrere Informationsbedürfnisse erfüllt werden können.

15 % der gestellten Anfragen waren leer, d.h. sie enthielten keine Anfrageterme. Von den gültigen Anfragen waren 68 % Anfragen, die eine neue Anfrage stellten und damit die erste Ergebnisseite anforderten. 32 % der gültigen Anfragen forderten eine weitere Ergebnisseite zu einer vorangegangenen Anfrage an. 4,2 % der gültigen Anfragen wiederholten sich exakt, d.h. sie beinhalteten genau dieselben Anfrageterme und forderten dieselbe Ergebnisseite an. Diese identischen Anfragen flossen nicht in die Auswertungen mit ein.

Eine durchschnittliche Anfrage hatte in der Altavista-Studie 2,35 Terme. In den Fällen, in denen eine Anfrage modifiziert wurde, wurden Terme hinzugefügt (7,1%), Terme gelöscht (3,1%) oder die Operatoren verändert (1,4%). In 35 % der Fälle veränderte sich die Anfrage komplett. Detailliertere Angaben zu den Ergebnissen der Altavista-Studie bezüglich Anfragelänge, Sitzungslänge und Ergebnisbetrachtung sind Tabelle 3.1 zu entnehmen.

Die Fireball-Studie [Höl00]: In dieser Studie wurde die deutsche Suchmaschine Fireball beobachtet. Die Studie betrachtet den Zeitraum Juni 1998.

In dieser Studie wurde wie in der Excite-Studie nicht unterschieden zwischen einer erneuten Suchanfrage mit den gleichen Termen und dem Anschauen einer weiteren Ergebnisseite. Hier wurden alle Fälle im Gegensatz zur Excite-Studie als erneute Anfrage betrachtet. Daher können bei der Fireball-Studie keine Angaben zur Ergebnisbetrachtung gemacht werden. Auch über die Länge einer Sitzung wird in dieser Studie nichts ausgesagt.

Eine Anfrage bestand in der Fireball-Studie aus durchschnittlich 1,66 Begriffen. Dieser Zahlwert ist deutlich geringer als bei den anderen beiden Studien. Der Unterschied mag sich dadurch erklären lassen, dass in der deutschen Sprache viele Begriffe und Konzepte als Komposita ausgedrückt werden, während diese im Englischen in Form von getrennten Wörtern formuliert werden.

Eine detailliertere Aufschlüsselung der Studienergebnisse kann man wiederum der nachfolgenden Tabelle 3.1 entnehmen.

	Fireball	**Excite**	**Altavista**
Zeitraum	30 Tage 1. bis 30.6.1998	1 Tag 9.3.1997	43 Tage 2.8. bis 13.9.1998
Indizierter Dokumentbestand[2]	6 Mio. Webseiten	40 Mio. Webseiten	140 Mio. Webseiten
Anzahl Anfragen	16 252 902	51 473	993 208 159
Anfrageterme pro Anfrage	Ø: 1,66 0: – 1: 54,59 % 2: 30,80 % 3: 10,36 % 4: 2,76 % 5: 0,94 % >5: 0,66 %	Ø: 2,21 0: 5 % 1: 31 % 2. 31 % 3: 18 % 4: 7 % 5: 4 % >5: 4 %	Ø : 2,35 0: 20,6 % 1: 25,8 % 2: 26,0 % 3: 15,0 % >3: 12,6 %
Anfragen pro Sitzung	Keine Angaben	Ø: 1,6 1: 67 % 2: 19 % 3: 7 % 4: 3 % >4: 4 %	Ø: 2,02 1: 77,6 % 2: 13,5 % 3: 4,4 % >3: 4,5 %
Verwendung von booleschen Operatoren	Keine: 63,43 % AND: 2,4 % OR: 0,09 % NOT: 0,06 % ( ): 0,1 % + : 24,82 % - : 0,48 % “ ”: 8,62 %	Keine: 75,93 % AND: 8 % OR: 0,34 % NOT: 0,2 % ( ): 0,53 % + : 6 % - : 3 % “ ” : 6 %	Keine: 79,6 %
Angeschaute Ergebnisseiten pro Anfrage	Keine Angaben	Ø: 2,35 1: 58 % 2: 19 % 3: 9 % 4: 5 % 5: 3 % >5: 6 %	Ø: 1,39 1: 85,2 % 2: 7,5 % 3: 3,0 % >3: 4,3 %

Tabelle 3.1: Übersicht über Studienergebnisse zur Benutzung von Suchmaschinen

[2] Zum Zeitpunkt der Studie

Noch einige Hinweise zur Interpretation der Ergebnisse: Die Vergleichbarkeit der Ergebnisse ist nur eingeschränkt möglich. Die Studien unterscheiden sich stark in der verwendeten Terminologie (request, query) und in der Untersuchungsmethode [JP00]. So wird beispielsweise eine Sitzung unterschiedlich identifiziert sowie das Anschauen der Ergebnisseiten unterschiedlich aus dem Datenmaterial hergeleitet. Weiterhin unterscheiden sich die Studien im Umgang mit leeren und „doppelten“ Anfragen, was sich auf die berechneten Zahlenwerte auswirkt. Weiterhin bieten die einzelnen Suchmaschinen dem Benutzer auch nicht genau dieselben Anfragemöglichkeiten und Optionen an.

Dennoch sind die Aussagen der drei Studien in etwa identisch. Die Sitzungen der Benutzer sind in der Regel recht kurz, am häufigsten bestehen Sitzungen aus einer Anfrage. Komplexere Formulierungen der Anfragen kommen in der Regel nicht vor, es werden meist nur wenige Terme verwendet und nur selten werden diese Terme mit logischen Verknüpfungen versehen. Zur Begutachtung der Ergebnisse wird meist nur die erste Seite betrachtet.

Die Altavista-Studie beschäftigt sich beispielsweise noch mit den Themen, nach denen am häufigsten gesucht wurde. Diese Themen kommen alle aus dem Freizeit- und Unterhaltungsbereich. Hierbei sind bekanntermaßen die Erwartungen hinsichtlich der Qualität der Dokumente bzw. Ergebnisse nicht so ausgeprägt wie bei Literaturrecherchen, die aus beruflichem Interesse durchgeführt werden und auf die Qualität von bei Verlagen veröffentlichten Dokumenten bauen. Von diesen Literaturrecherchen werden die Studien im nächsten Abschnitt berichten.

3.2.2.2 Benutzerstudien von Literaturdiensten im WWW

Im Folgenden werden drei Benutzerstudien vorgestellt und miteinander verglichen, die sich an eine speziellere Benutzergruppe richten: an Wissenschaftler im Bereich der Informatik und fortgeschrittene Informatik-Studenten, die in Literaturdiensten recherchieren. Insofern decken diese Studien einige Randbedingungen mehr ab als die vorangegangenen. Die Benutzer suchen für ihre berufliche Arbeit und im Bereich der Informatik sollte ein gewisses Geschick im Umgang mit den Systemen vorausgesetzt werden können. Das Dokumentmaterial ist von höherer Qualität als allgemeine Webseiten. Die Dienste sind alle über eine Webschnittstelle für die Allgemeinheit zugänglich. Vorgestellt werden Studien an den Diensten: New Zealand Digital Library (NZDL), Informatik-Bibliographie (CSBIB) und ResearchIndex (RI).

New Zealand Digital Library: In [JCM98] und [JCMB00] werden die Ergebnisse einer Langzeitstudie zur Benutzung der NZDL beschrieben. NZDL umfasst zahlreiche Dokumentensammlungen zu unterschiedlichen Themen, u.a. eine Sammlung von Technischen Berichten aus dem Bereich der Informatik, die Gegenstand des vorliegenden Experiments war. Zum Zeitpunkt der Untersuchungen waren etwa 46 000 Technische Berichte verfügbar. Alle wurden mit ihrem gesamten Volltext indiziert.

Eine Sammlung von Technischen Berichten ist eher mit einem traditionellen OPAC zu vergleichen als mit einer allgemeinen Suchmaschine. Allerdings bietet dieser Dienst nicht die üblichen Anfragefelder wie Titel, Autor oder Schlagwort an. Dafür kann man die Suche jeweils auf die erste Seite der Dokumente beschränken, so dass man dadurch sozusagen die Suche im Titel, den Autoren und der Zusammenfassung nachbilden kann. Anfragen können mit booleschen Operatoren formuliert werden oder nur durch eine Menge von Stichworten, dann wird das Ergebnis gemäß der Ähnlichkeit zu diesen Worten sortiert. Die Benutzer der NZDL stammten hauptsächlich aus dem wissenschaftlichen Umfeld, insbesondere aus dem Bereich der Informatik. Die Benutzerdaten wurden über einen Zeitraum von 61 Wochen erhoben, in dieser Zeit erfolgten über 30 000 Anfragen.

Etwa 80 % der NZDL-Anfragen bestanden aus drei oder weniger Termen. Die durchschnittliche Termanzahl wurde in der Studie nicht angegeben. Bei 66 % der Anfragen wurden keine booleschen Operatoren verwendet. Detailliertere Aufschlüsselungen der Ergebnisse in Bezug auf Anfragelänge, Sitzungslänge und Verwendung der booleschen Operatoren findet man in Tabelle 3.2.

Der NZDL-Dienst präsentiert standardmäßig 25 Treffer auf einer Ergebnisseite. Der Benutzer kann allerdings bei der Anfrage die Menge der gewünschten Treffer auf 100, 200 oder 500 Treffer vergrößern. In 90 % der Anfragen wurde die Standardgröße beibehalten. Die mittleren Ergebnisgrößen wurden jeweils in 2,5 % der Anfragen gewählt, 500 Ergebnisse wurden in knapp 6 % der Anfragen gefordert.

In 64,2 % der Recherchen haben die Benutzer kein einziges Dokument genauer angesehen, d.h. den Volltext abgerufen. Auf der Ergebnisseite wird allerdings auch zu jedem Dokument die Kurzfassung angegeben. Genau ein Volltext wurde in 19,2 % der Sitzungen betrachtet. Zwei Dokumente wurden in 7 % der Recherchen inspiziert und in lediglich 4 % der Recherchen wurden 5 oder mehr Volltexte angeschaut. 73,2 % der betrachteten Dokumente befanden sich auf den ersten 25 Plätzen der Ergebnisliste, am häufigsten wurde das Dokument auf Platz 1 angeschaut.

Die Anfrageschnittstelle von NZDL gibt bereits viele Standardeinstellungen vor, wie beispielsweise die Art der Anfrage (boolesch oder gemäß des Vektorraummodells), die Nachbarschaft von Worten, die Verwendung von Wortstammreduktion, die Berücksichtigung von Groß-/Kleinschreibung oder die Größe der Ergebnismenge. Die Vorgaben dieser Einstellungen wurden im Laufe des Experiments mehrmals verändert. Zwei Drittel der Benutzer übernahmen die vorgegebenen Einstellungen, unabhängig davon wie diese gesetzt waren. 21 % der Benutzer änderten die Nachbarschaftsbedingung der Worte, 10 % der Benutzer nahmen Einfluss auf die Ergebnismengengröße. In 6 % der Sitzungen machten Benutzer Gebrauch von der Online-Hilfe.

Bei der manuellen Analyse der Benutzungsdaten fiel auf, dass etlichen Besuchern nicht klar war, welche Dokumente sich im NZDL-Dienst verbergen, nämlich Technische Berichte im

Bereich der Informatik. Die Anfragen zielten teilweise auf andere Themen und Bereiche ab, teilweise wurde nach einer ISBN gesucht, die gerade eben nicht für Technische Berichte vergeben wird (graue Literatur).

Informatik-Bibliographie: In [MC00] werden die Ergebnisse einer Benutzerstudie an der Informatikbibliographie (Collection of Computer Science Bibliographies, CSBIB) präsentiert. Der CSBIB-Dienst beinhaltet mehr als 1 Million Dokumentreferenzen, hauptsächlich von Konferenzartikeln, Zeitschriftenbeiträgen und Technischen Berichten aus dem Bereich der Informatik. CSBIB hält hauptsächlich die bibliographischen Daten vor, nur vereinzelt sind auch die Volltexte abrufbar. Die Benutzerdaten wurden während eines Zeitraums von knapp 4 Monaten mitprotokolliert. Dabei kam es zu mehr als 250 000 Anfragen.

Durchschnittswerte über die Anzahl der Anfrageterme oder die Sitzungslänge werden in der Studie nicht angegeben. Detailliertere Aufschlüsselungen der Ergebnisse in Bezug auf Anfragelänge, Sitzungslänge und Verwendung der booleschen Operatoren sind in Tabelle 3.2 zu finden.

Darüber hinaus wurde beobachtet, dass 72,48 % der Benutzer den CSBIB-Dienst nur ein einziges Mal im Erhebungszeitraum besucht haben. 11,24 % der Benutzer wendeten sich immerhin zweimal an CSBIB und nur 9,02 % der Benutzer benutzten den Dienst 5 Mal oder mehr.

ResearchIndex: In [MC01] wird eine TLA-Benutzerstudie von ResearchIndex vorgestellt. Zum Zeitpunkt der Datenerhebung umfasste der Dienst mehr als 290 000 Volltext-Dokumente, hauptsächlich Konferenz- oder Zeitschriftenartikel aus dem Bereich der Informatik und Elektrotechnik. Die Dokumente wurden automatisch im WWW aufgespürt und in verschiedenen Formaten (z.B. .ps, .pdf) archiviert. Zunächst wurde der gesamte Textkörper herausgefiltert und daraus wurden wiederum sowohl die bibliographischen Angaben als auch die enthaltenen Literaturreferenzen automatisch extrahiert. Diese charakteristischen Textteile eines wissenschaftlichen Artikels wurden separat indiziert und nutzbar gemacht. Aufgrund der Verknüpfung der Dokumente gemäß ihrer Zitiert-Beziehungen erfreut sich ResearchIndex einer großen Beliebtheit und liefert einen echten Mehrwert gegenüber anderen Diensten.

Bei ResearchIndex ist eine boolesche Suche im Volltext- und im Referenzen-Index möglich, eine Suche über ausgewiesene Felder existiert nicht. Für detailliertere Aufschlüsselungen der Untersuchungsergebnisse in Bezug auf Anfragelänge, Sitzungslänge und Verwendung der booleschen Operatoren sei wiederum auf Tabelle 3.2 verwiesen.

Zusätzlich wurde die Zeitdauer einer Sitzung untersucht. 57,47 % der Sitzungen dauerten eine Minute oder kürzer. Für 6,48 % der Sitzungen betrug die Dauer zwischen einer und zwei Minuten, für 15 % der Sitzungen lag die Zeit zwischen 10 und 30 Minuten.

	NZDL	**CSBIB**	**ResearchIndex**
Zeitraum	61 Wochen Apr ´96 - Juli ´97	17 Wochen Sept ´99 - Dez ´99	29 Wochen Aug ´99 - Feb ´00
Indizierter Dokumentbestand[3]	46 000 Technische Berichte im Volltext	> 1 000 000 Bibliogr. Daten von Artikeln	290 000 Artikel im Volltext
Anzahl Anfragen pro Woche	428	14 816	53 143
Suchschnittstelle	Volltext-Index Stichwortsuche Boolesch/Ranked	Suchfelder Boolesche Suche	Volltext-Index Stichwortsuche Boolesch/Ranked
Anzahl Anfragen	32 802	251 878	1 541 148
Anzahl Sitzungen	26 128	54 671	46 486
Anfrageterme pro Anfrage	1: 27,06 % 2: 34,04 % 3: 19,76 % 4: 8,98 % 5: 4,26 % 6: 2,06 % >6: 2,25 %	1: 52,72 % 2. 28,10 % 3: 10,80 % 4: 4,18 % 5: 1,75 % 6: 1,02 % >6: 1,41 %	1: 37,02 % 2: 30,98 % 3: 12,44 % 4: 13,34 % 5: 2,16 % 6: 1,37 % >6: 2,67 %
Anfragen pro Sitzung	1: 43,89 % 2: 21,95 % 3: 12,10 % 4: 7,76 % 5: 4,88 % 6: 2,90 % 7: 1,92 % 8: 1,53 % >8: 2,41 %	1: 35,97 % 2: 20,02 % 3: 12,19 % 4: 8,51 % 5: 5,84 % 6: 3,83 % 7: 2,68 % 8: 2,13 % >8: 8,81 %	1: 46,69 % 2: 11,51 % 3: 7,48 % 4: 5,42 % 5: 3,86 % 6: 3,10 % 7: 2,63 % 8: 2,07 % >8: 17,17 %
Verwendung von Booleschen Operatoren	Keine: 66,0 % AND: 25,8 % OR: 2,5 % ( ): 4,6 %	Keine: 84,10 % AND: 14,18 % OR: 1,96 % ( ): 0,01 %	Keine: 62,42 % AND: 14,73 % OR: 12,78 % ( ): 0,75 % NEAR: 9,32 %

Tabelle 3.2: Übersicht über Studienergebnisse zur Benutzung von Literaturkatalogen

[3] Zum Zeitpunkt der Studie

Auch hier noch einige Hinweise zur Interpretation der Ergebnisse: Die durchgeführten Studien zur Literaturrecherche sind jeweils sehr speziell und damit in ihrer allgemeinen Aussagefähigkeit sicherlich begrenzt. Wie bei den TLA-Studien der Suchmaschinen ist die Vergleichbarkeit der hier zusammengestellten Ergebnisse auch nur eingeschränkt möglich. Die Kürze der Darstellung spiegelt nicht alle Annahmen und Voraussetzungen der einzelnen Studien wider, auch bei diesen Studien wurden beispielweise Sitzungen und leere Anfragen unterschiedlich behandelt. Für zusätzliche Informationen sei auf die zitierten Artikel verwiesen. Im Großen und Ganzen ähneln und bestärken sich die Ergebnisse der Studien gegenseitig. Für die Zwecke der vorliegenden Arbeit, nämlich um eher qualitative Aussagen über das Benutzerverhalten gewinnen zu können, sind die gefundenen Ergebnisse hinreichend valide und durchaus aufschlussreich.

Bewertung

Vergleicht man die Untersuchungsergebnisse der allgemeinen Suchmaschinen mit denen der Literaturrecherchedienste, fällt auf, dass das Verhalten bei fachkundigen Personen, d.h. wissenschaftlich arbeitenden Informatikern, nicht wesentlich anders ausfällt als bei den Benutzern einer Suchmaschine. Auch bei der Literaturrecherche sind längere Sitzungen äußerst selten, sowohl was die Menge der Folgeanfragen als auch was die verwendete Zeit betrifft. Die Zahl der Anfrageterme ist ebenfalls recht gering. Sogar die Verwendung von booleschen Operatoren ist recht verhalten, zumal man bei Informatikern annehmen kann, dass ihnen die Bedeutung und Handhabung vertraut ist. Die Suchergebnisse werden in beiden Anwendungsbereichen keineswegs ausgiebig betrachtet und studiert, vielmehr werden einige wenige Treffer inspiziert und davon auch eher die, die weit oben in der Ergebnisliste aufgeführt sind.

Trotz der niedrigen Durchschnittswerte bezüglich der Anfrageterme und der Sitzungslänge macht die detaillierte Aufschlüsselung der Ergebnisse aber auch deutlich, wie groß die Varianz in diesen Bereichen ist. D.h. es ist immer auch mit (einer Minderheit von) Benutzern zu rechnen, die komplexere Anfragen formulieren und längere Recherchesitzungen durchführen, wie beispielsweise bei der ResearchIndex-Studie berichtet wurde. Ebenso gab es auch Benutzer der NZDL, die trotz anderslautender Vorgaben 500 Ergebnisse angefordert haben.

3.2.3 Resümee

Das Thema „Benutzerverhalten bei der Informationssuche“ wurde in diesem Abschnitt aus zwei verschiedenen Blickwinkeln betrachtet.

Zum einen wurde das Informationssuchverhalten des Benutzers durch das Aufstellen von Prozessmodellen und im Zusammenhang mit Lerntheorien interpretiert. Dadurch konnte ein Bild gewonnen werden von den verschiedenen Zuständen, inneren und äußeren Vorgängen und vor allem von den Schwierigkeiten, denen sich der Benutzer bei einer Informationssuche

gegenüber sieht. Sicherlich ist es ein sinnvolles Ziel, die Informationssuche als Prozess zu unterstützen. Es wurden allerdings auch die Schwierigkeiten und Grenzen einer technischen Unterstützung klar, da viele Phasen oder Vorgänge des Prozesses im Benutzer selbst und durch Kommunikation mit anderen Personen – also außerhalb des Systems – stattfinden.

Zum anderen wurden in TL-Analysen die konkreten Aktionen der Benutzer beobachtet. Anhand der Untersuchungsergebnisse konnte der Prozesscharakter der Informationssuche zum jetzigen Zeitpunkt jedoch noch nicht im Benutzerverhalten deutlich werden. Die beobachteten Sitzungen waren recht kurz, die Anfragen enthielten nur wenige Terme, auch wurden kaum Veränderungen an der Anfrageformulierung vorgenommen. Zeit und Aufwand, die ein Benutzer für eine Informationssuche für angemessen hält, erwiesen sich als sehr gering.

Anhand der dadurch gewonnenen Eindrücke und Erkenntnisse soll nun Anforderung A4 des eingangs formulierten Anforderungskatalogs, nämlich die Berücksichtigung menschlicher Fähigkeiten, Fertigkeiten und Grenzen konkretisiert werden.

Dem langandauernden und iterativen Prozesscharakter der Informationssuche kann beispielsweise dadurch Rechnung getragen werden, dass das *Erinnerungsvermögen* des Benutzers unterstützt wird. So sollten zumindest frühere Anfragen eingesehen werden können, vielleicht auch bereits bekannte Dokumente speziell markiert werden.

Bei der Betrachtung des Suchprozesses wurde gleichzeitig auch der Einfluss der *Individualität* des Benutzers deutlich. Das Handeln des Benutzers ist in erster Linie von seinen persönlichen Vorlieben und Fertigkeiten geprägt und unterliegt gleichzeitig auch situativen Gegebenheiten und emotionalen Schwankungen. Daher sollten vom System möglichst wenig Vorannahmen getroffen werden, sondern eher eine Reihe von Alternativen angeboten werden, die jeder Benutzer dann je nach Bedarf individuell nutzen kann. Die TLA-Studien haben ebenfalls gezeigt, mit welcher *Bandbreite* bei den Benutzern zu rechnen ist. Auch wenn die Mehrheit der Benutzer keine langen Recherchesitzungen durchführt, gibt es dennoch einen gewissen Anteil an Benutzern, der diese Möglichkeiten ausnutzt und schätzt.

Im allgemeinen hat sich allerdings bei den Studien auch gezeigt, dass die beim Benutzer zu erwartenden *Fähigkeiten und Fertigkeiten* bei der Durchführung einer Informationssuche eher gering einzuschätzen sind, selbst bei technisch vorgebildeten Informatikern.

Schließlich hat sich gezeigt, wie wenig Zeit und Aufwand die Mehrheit der Benutzer für die Suche und Ergebnisbetrachtung bereit ist einzusetzen. Wir wollen das hier als *Ungeduld* bezeichnen und meinen damit, dass sich die meisten Benutzer nur ungern lange mit der Formulierung einer Anfrage abgeben und auch die Ergebnisse eher nach der Schnelle der Verfügbarkeit beurteilen als dass sie sich die Zeit für eine eingehendere Beurteilung bzw. eine optimale Entscheidung nehmen.

Damit wollen wir A4 durch folgende Aspekte konkretisieren:

- Erinnerungshilfen anbieten
- Individualität ermöglichen
- Bandbreite zulassen
- niedrige Erwartungen bezüglich Suchexpertise stellen
- von der Ungeduld des Benutzers ausgehen

Diese Aufzählung ist sicherlich nicht vollständig. Sie umfasst allerdings eine Reihe von kritischen Aspekten, die – stellvertretend für die im gesamten Abschnitt gewonnenen Eindrücke und Erkenntnisse – im Folgenden wichtige Anhaltspunkte bei der Beurteilung und technischen Gestaltung von Recherchewerkzeugen liefern können.

3.3 Existierende Meta-Recherchesysteme

In diesem Abschnitt stellen wir existierende Meta-Recherchesysteme vor und bewerten sie im Hinblick auf die Erfüllung unseres Anforderungskatalogs. Zunächst betrachten wir Meta-Suchmaschinen für das WWW, da das WWW sozusagen das erste Anwendungsfeld war, an dem der durchschlagende Erfolg von Meta-Rechercheansätzen sichtbar wurde. Anschließend betrachten wir existierende Meta-Recherchesysteme für den Bereich der wissenschaftlichen Literatursuche.

3.3.1 Meta-Suchmaschinen für das WWW

Nachdem im Juli 2000 die Zwei-Milliarden-Marke an HTML-Seiten durchbrochen wurde, gehen heutige Schätzungen (Stand: Anfang 2002) von 10 Milliarden existierenden HTML-Seiten aus. Davon können von einzelnen Suchmaschinen natürlich nur Bruchteile der vorhandenen Informationsmenge indiziert und suchbar gemacht werden. Tabelle 3.3 zeigt die vom Betreiber selbst angegebene Größe und eine durch Tests berechnete Größe der größten Suchmaschinen. Mit Größe (engl. size) ist die Anzahl der von einer Suchmaschine indizierten Webseiten gemeint. Folgende Tabelle, genauso wie Tabelle 3.4, ist dem Search Engine Showdown von Greg R. Notess (zu finden unter http://www.searchengineshowdown.com) entnommen. Dort werden zahlreiche Informationen über aktuelle Entwicklungen im Bereich der Suchmaschinen für interessierte Benutzer zusammengestellt. Die Informationen betreffen beispielsweise neue Suchfunktionalitäten, liefern detaillierte Bewertungen und Vergleiche von diversen Suchmaschinen und berichten von statistischen Analysen und durchgeführten Tests, wie z.B. der hier angeführten Showdown Schätzung über die Zahl der indizierten HTML-Seiten.

Suchmaschine	Showdown Schätzung (in Millionen)	Angabe des Betreibers (in Millionen)
Google	968	1 500
WiseNut	579	1 500
AllTheWeb	580	507
Northern Light	417	358
AltaVista	397	500
Hotbot	332	500
MSN Search	292	500

Tabelle 3.3: Größe der Suchmaschinen (Stand März 2002)

Zudem gibt es auch Studien über den Grad der Überschneidung zwischen den einzelnen Suchmaschinen. Showdown führt dazu regelmäßig Experimente durch, in welchen jeweils die größten Suchmaschinen mit einer Menge von vier bis fünf Anfragen befragt werden. Die gesamte Treffermenge wird dann dahingehend untersucht, wie viele Webseiten von einer, von zwei, von drei, usw. Suchmaschinen gefunden werden können. Tabelle 3.4 stellt einige Informationen dazu zusammen. In Anhang B sind ausführlichere Angaben zu den Showdown Experimenten zu finden. Die Experimente zeigen, dass ein Großteil der gefundenen Webseiten nur von einer einzigen Suchmaschine gefunden wurde. Nimmt man die von bis zu drei Suchmaschinen gefundenen Webseiten zusammen, so machen diese prozentual jeweils schon einen Anteil von mehr als 75 % der Gesamttreffer aus. Folglich werden weniger als 25 % der gefundenen Webseiten von mehr als drei Suchmaschinen nachgewiesen. D.h. in Anbetracht von zehn und mehr befragten Suchmaschinen ist der Überlappungsgrad dieser Suchmaschinen als ziemlich gering zu bewerten.

Zeitpunkt	# befragter Such-maschinen	Insgesamt gefundene Webseiten	Nur einmal gefundene Webseiten	Zweimal gefundene Webseiten	Dreimal gefundene Webseiten
März 2002	10	141 (100%)	71 (50,4%)	30 (21,3%)	10 (7,1%)
Februar 2000	14	298 (100%)	110 (36,9%)	79 (26,5%)	40 (13,4%)
September 1999	13	140 (100%)	66 (47,1%)	30 (21,4%)	22 (15,7%)
Mai 1999	11	122 (100%)	63 (51,6%)	24 (19,7%)	6 (4,9%)

Tabelle 3.4: Überschneidungen zwischen Suchmaschinen

Diese Zahlen machen aber auch der Gewinn einer Zusammenführung von einzelnen Suchdiensten zu einem Meta-Recherchesystem offensichtlich. Mit Meta-Suchmaschinen können deutlich mehr Webseiten gefunden werden als mit einzelnen Suchmaschinen, obwohl diese Einzeldienste auch schon enorme Mengen an Webseiten indiziert haben. Angesichts der vielen nur einmal gefundenen Webseiten sollte eine Meta-Suchmaschine also durchaus zehn oder mehr integrierte Einzeldienste enthalten, wenn sie dem Benutzer eine umfassende Recherchemöglichkeit bieten will.

Meta-Suchmaschinen führen keinen eigenen Datenbestand, sie setzen auf den etablierten Suchmaschinen auf, transformieren die Anfragen der Benutzer, so dass sie von den integrierten Suchmaschinen verstanden werden, führen eine parallele Befragung dieser Suchmaschinen durch und sammeln die Ergebnisse auf. Nach einer Transformation in ein einheitliches Ergebnisformat werden dem Benutzer die Treffer präsentiert. Meta-Suchmaschinen sind also besonders dann geeignet, wenn ein Benutzer eine umfassende Recherche durchführen möchte oder wenn er nach eher seltenen Informationen sucht.

Im Folgenden werden vier bekannte Meta-Suchmaschinen etwas genauer beschrieben.

MetaGer. MetaGer ist eine Meta-Suchmaschine, die hauptsächlich interessante Suchdienste für den deutschsprachigen Raum integriert. Für eine Anfrage kann der Benutzer die Terme mit AND, OR und NOT verknüpfen, eine Phrasensuche wird nur für den Titel und die Zusammenfassung einer HTML-Seite unterstützt. MetaGer integriert 21 Suchmaschinen (Stand: Dezember 2002), die vom Benutzer für jede Suche individuell kombiniert werden können. Den Suchvorgang kann der Benutzer durch die spendierte Suchzeit bei jedem Dienst und die insgesamt gewünschte Trefferzahl beeinflussen. Standardmäßig sucht MetaGer in acht vorausgewählten Diensten für die Zeitdauer von 10 Sekunden und gibt als Gesamtergebnis maximal 200 Treffer zurück. Zu Beginn der Ergebnismenge wird eine Statistik über die gefundene Trefferanzahl pro Suchmaschine angezeigt. Duplikate werden in der Ergebnismenge grundsätzlich erkannt und dem Benutzer deutlich gemacht.

Für die Ergebnisanzeige kann der Benutzer auswählen, ob er die Treffer mit wenig, mäßig und viel zusätzlichem Text ansehen möchte. Weiterhin kann er zwischen verschiedenen Ergebnisstrukturen wählen. MetaGer bietet die Sortierung nach Relevanz und die Gruppierung der Ergebnisse nach Webdomänen an. Darüber hinaus kann der Benutzer die Treffer vor der Präsentation in der Ergebnismenge auf Existenz testen lassen. Dadurch entsteht eine aktuelle und bereinigte Ergebnisanzeige, allerdings dauert diese Form der Präsentation auch wesentlich länger. Zur Überbrückung der normalen und auch dieser verlängerten Wartezeiten wird der Benutzer mit wechselnden amüsanten Lebensweisheiten unterhalten.

MetaCrawler. MetaCrawler ist eine seit langem etablierte Meta-Suchmaschine für internationale Suchen. Sie wurde 1994 an der Universität von Washington entwickelt und wird seit Juli 2000 von der Firma InfoSpace Inc. betrieben. MetaCrawler erlaubt die

Verknüpfung von Anfragetermen mit AND und OR sowie die Formulierung von Phrasen. Zur Zeit (Dezember 2002) integriert MetaCrawler zehn Suchdienste. Zur Steuerung des Anfragevorgangs kann der Benutzer entweder nur die schnellsten Dienste befragen oder eine Zeitschranke von maximal zwei Minuten angeben, in der jeder Dienst seine Ergebnisse liefern kann. Der Benutzer kann sowohl die Treffer pro Suchdienst als auch die auf einer Seite präsentierten Ergebnisse beeinflussen, allerdings sind jeweils nur Mengengrößen zwischen 10 und 30 erlaubt. Vor der Ergebnisanzeige gibt MetaCrawler die befragten Suchdienste und die Anzahl der von ihnen bezogenen Treffer an. MetaCrawler bietet für die Präsentation der Ergebnistreffer verschiedene Sortierungen an, nach Relevanz der Seite, nach der Domäne oder nach dem Suchdienst, durch den sie gefunden wurde. Duplikate werden stets zusammengeführt und für den Benutzer kenntlich gemacht. Über Cookies kann der Benutzer die individuellen Einstellungen einer Suche für eine spätere Wiederverwendung speichern.

ProFusion. ProFusion bietet zur Formulierung von Anfragen die AND-, OR- und die Phrasen-Option an. ProFusion bindet 13 Suchmaschinen ein (Dezember 2002). Zur Steuerung des Anfragevorgangs kann der Benutzer die schnellsten fünf oder alle Dienste befragen, er kann aber auch manuell eine individuelle Auswahl der Suchdienste zusammenstellen. Die maximale Obergrenze für die Suchzeit beträgt bei ProFusion zwei Minuten. Bevor die Ergebnisse angezeigt werden, informiert ProFusion den Benutzer durch Fortschrittsbalken über den Verlauf seiner Suche, d.h. welche Dienste zum jeweiligen Zeitpunkt bereits wie viele Ergebnisse geliefert haben. Pro Suchdienst berücksichtigt ProFusion zwischen 10 und 30 Treffer, die Ergebnisseite kann der Benutzer auch so groß wählen, dass alle Treffer auf einer Seite angezeigt werden können. Duplikate werden stets berechnet und für den Benutzer sichtbar gemacht, ungültige Links werden entfernt und tauchen nicht im Ergebnis auf. Der Benutzer kann eine Ergebnisstatistik einsehen, die die Duplikate und ungültigen Links genauer aufschlüsselt. ProFusion bietet verschiedene Sortierungen für die Ergebnisse an, nach Relevanz, Titel der Seite, URL oder der Suchmaschine, die sie gefunden hat. ProFusion erlaubt es dem Benutzer, sich bei neuen Ergebnissen oder Änderungen an bestimmten Seiten benachrichtigen zu lassen und die gewählten Sucheinstellungen für spätere Anfragen zu speichern.

Vivisimo. Vivisimo ist nicht nur eine Meta-Suchmaschine, sondern auch eine automatische Clustering-Maschine. Dadurch bietet Vivisimo dem Benutzer eine zusätzliche Unterstützung im Umgang mit großen Ergebnismengen. Sie erlaubt die Formulierung von Anfragen mit AND, OR und Phrasen und unterstützt sogar die Suche in separaten Feldern wie z.B. im Titel, der Domäne oder der URL einer Seite. Vivisimo integriert fünf allgemeine Suchmaschinen (Dezember 2002) und zusätzlich auch spezialisierte Suchmaschinen für Nachrichten oder kommerzielle Angebote. Der Benutzer kann bis zu 30 Sekunden Wartezeit für sein Ergebnis spendieren. Vivisimo kann mit Treffermengen bis zu 500 Treffern umgehen und die gefundenen Dokumente in bedeutungsvollen Gruppen anordnen. Diese Gruppierung geschieht anhand von gemeinsamen Begriffen in der gelieferten Kurzfassung. So entsteht ein hierarchisches semantisches Netz aus Gruppen, das vollständig dynamisch generiert wird. Ein

Dokument kann durchaus in mehreren Gruppen auftauchen. Vivisimo verzichtet also bewusst auf ein eindimensionales Rangieren. Duplikate werden stets berechnet und zusammengeführt, der Benutzer kann beeinflussen, welche Informationen ihm zu jedem Treffer angezeigt werden, er kann zwischen der URL, einer Kurzfassung oder dem Anfang einer Seite wählen.

Mittlerweile existieren eine ganze Reihe von Meta-Suchmaschinen für das WWW, vgl. hierzu http://www.searchenginewatch.com/links/metacrawlers.html. Zur Bewertung und zum Vergleich von Meta-Suchmaschinen haben Forscher einen Kriterienkatalog (K1-7) für die Güte und Qualität einer Meta-Suchmaschine formuliert [BS98]:

- **Parallele Suche:** Damit werden „all-in-one"-Formulare ausgeschlossen. Eine Meta-Suchmaschine muss wirklich parallel, d.h. gleichzeitig in mehreren verschiedenen Suchdiensten suchen (K1).
- **Ergebniszusammenführung:** Die Ergebnisse der einzelnen Suchdienste müssen zusammengeführt, vermischt und in einem einheitlichen Format dargestellt werden (K2).
- **Duplikateliminierung:** Doppelte Fundstellen müssen erkannt und gekennzeichnet werden (K3).
- **Mindestens AND- und OR-Operatoren:** Als logische Operationen müssen mindestens die Operatoren AND und OR für die Anfrageformulierung zur Verfügung stehen, die Unterstützung von Phrasen ist allerdings ebenso wünschenswert (K4).
- **Kein Informationsverlust:** Wenn ein Suchdienst beispielsweise eine Kurzbeschreibung bzw. den Anfang einer HTML-Seite als Ergebnis liefert, dann müssen diese Informationen auch in das Gesamtergebnis übernommen werden (K5).
- **Transparenz:** Die spezifischen Eigenschaften der eingebundenen Suchdienste dürfen für die Bedienung der Meta-Suchmaschine keine Rolle spielen, der Benutzer muss nichts darüber wissen müssen (K6).
- **Vollständige Suche:** Die Meta-Suchmaschine sollte in der Lage sein, so lange zu suchen, wie irgendeiner der eingebundenen Suchdienste noch Treffer liefert (K7).

Diese Kriterien decken sich recht gut mit unseren zu Beginn des Kapitels formulierten Anforderungen und unterstützen diese dadurch noch einmal. Genau genommen stellen die genannten Kriterien eine Verfeinerung unserer Anforderungen dar und liefern Unterpunkte zu den Anforderungen A1 (K1, K2, K6), A3 (K4), A5 (K5, K7) und A6 (K3).

Tabelle 3.5 gibt einen Überblick über die Erfüllung der genannten Kriterien durch bekannte Meta-Suchmaschinen. Die Darstellung ist entnommen aus [San98] und spiegelt den Stand von 1998 wider. Die Tabelle enthält eine Reihe von Suchmaschinen, die wir hier nicht weiter im Detail vorstellen wollen. Ergänzt haben wir noch die Meta-Suchmaschine Vivisimo, die es zum Zeitpunkt der Tabellenerstellung noch nicht gab. Daher wurde eine aktuelle Version (Stand Dezember 2002) von Vivisimo nach den vorgegebenen Kriterien bewertet.

Tabelle 3.5 macht deutlich, dass die Ergebnisintegration und die Duplikateliminierung nur von einem Teil der Dienste unterstützt wird, eine vollständige Suche ist sogar nur bei wenigen Diensten möglich. Die Anforderungen in Bezug auf eine parallele Suche, die Anfragefunktionalität und die Transparenz werden von der Mehrheit der Meta-Suchmaschinen erfüllt. Die Transparenz von ProFusion ist in der aktuellen Version mittlerweile gewährleistet.

Meta-Suchmaschine	Parallele Suche	Ergebnis-integration	Duplikat-eliminierung	AND/OR		Trans-parenz	Vollst. Suche
Cyber 411	v	–	–	–	–	v	–
DigiSearch	v	–	–	v	v	–	–
Dogpile	v	–	–	v	–	v	–
Highway 61	v	v	v	v	v	v	v
Inference Find	v	teilweise	v	v	–	–	–
Mamma	v	v	–	v	v	v	–
MESA	v	v	v	v	–	v	–
MetaCrawler	v	v	v	v	v	v	–
MetaGer	v	v	v	v	v	v	v
Metasearch	v	–	–	–	–	–	–
ProFusion	v	v	v	v	v	–	–
SavvySearch	v	–	–	v	v	v	–
Verio	v	v	v	–	–	v	–
Vivisimo	v	v	v	v	v	v	–
v vorhanden; – nicht vorhanden							

Tabelle 3.5: Übersicht über die Funktionalität bekannter Meta-Suchmaschinen

Von den detaillierter vorgestellten Meta-Suchmaschinen ist laut Tabelle lediglich MetaGer in der Lage, eine vollständige Suche durchzuführen. Das ist für die aktuelle Version von MetaGer nicht ganz korrekt. MetaGer erlaubt zwar eine Menge von 10 Millionen gefundenen Treffern (was sicherlich so gut wie unbegrenzt ist), allerdings liegt das Zeitlimit hierfür bei maximal 200 Sekunden. Im Vergleich dazu erlauben MetaCrawler und ProFusion nur 30 Treffer pro Suchdienst in weniger als 2 Minuten und Vivisimo lässt insgesamt 500 Treffer zu, sofern diese in 30 Sekunden gefunden werden konnten.

Bewertung

A1 (Einheitliche, integrierte Benutzungsschnittstelle): Eine einheitliche und integrierte Benutzungsschnittstelle ist bei allen hier vorgestellten Meta-Suchmaschinen vorhanden. Bei MetaGer und Vivisimo geht die Integration sogar noch etwas weiter, hier kann der Benutzer angeben, wie viel Informationen zu den gefundenen Webseiten (Kurzfassung, Seitenanfang) im Gesamtergebnis integriert sein sollen.

A2 (Einfache, intuitive Benutzerinteraktion): Eine einfache und intuitive Benutzerinteraktion ist ebenso bei den genannten Meta-Suchmaschinen vorhanden. Der Benutzer findet jeweils ein gewohntes HTML-Suchformular vor und kann sofort mit einer Stichwortsuche beginnen. Von der Vielzahl der angebotenen Optionen braucht er keine einzige aktiv auszuwählen, die Optionen sind mit geeigneten Standardwerten vorbelegt. Jede Meta-Suchmaschine bietet sowohl ein einfaches Suchformular an, das die möglichen Anfrageoptionen verbirgt, als auch eine Anfrageschnittstelle für Fortgeschrittene, die dann die Konfiguration aller angebotenen Anfrageoptionen erlaubt.

A3 (Ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit): Eine ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit ist bei den vorgestellten Meta-Suchmaschinen vorhanden. Die Anfrageschnittstelle für Fortgeschrittene erlaubt es dem Benutzer, zahlreiche Optionen zur Steuerung des Anfragevorgangs selbst zu bestimmen. Beispielsweise kann der Benutzer bei jeder Meta-Suchmaschine die anzufragenden Suchdienste explizit zusammenstellen. Die Optionen zur Festlegung der Ergebnismengengröße und zur Wartezeit sind allerdings nicht frei wählbar, d.h. der Benutzer kann lediglich eine von verschiedenen vorgegebenen Belegungen auswählen. Die vorgestellten Meta-Suchmaschinen bieten Strukturierungshilfen für die Begutachtung der Ergebnismenge an. Gerade bei größeren Ergebnismengen, die bei Meta-Suchmaschinen häufig vorkommen, ist eine solche Unterstützung sehr wichtig. Angeboten werden meistens verschiedene Sortierkriterien wie thematische Relevanz oder Domänenzugehörigkeit. Ganz im Trend des Semantic Web kann man die Treffer nun auch konzeptuell ordnen lassen (Vivisimo). MetaGer und MetaCrawler geben dem Benutzer eine kurze Statistik über die Anzahl der gefundenen Treffer, ProFusion bietet zu jedem Suchergebnis eine detaillierte Statistik über Duplikate sowie gefundene ungültige Links an. Zusätzlich kann der Benutzer bei ProFusion über Fortschrittsbalken den Zustand und Verlauf seiner Suche verfolgen.

A4 (Berücksichtigung menschlicher Eigenschaften): MetaCrawler und ProFusion erlauben es dem Benutzer, die gewählten Suchoptionen zu speichern und so bei Bedarf für spätere Recherchen wiederverwenden zu können. ProFusion unterstützt das Gedächtnis des Benutzers sogar noch soweit, dass er sich über Änderungen an bestimmten Seiten oder am Suchergebnis einer Anfrage informieren lassen kann. Die Ungeduld des Benutzers wurde von allen vorgestellten Systemen berücksichtigt. Sie bieten jeweils die Option an, nur die schnellsten Dienste zu befragen. MetaGer versucht sogar noch, dem Benutzer die Wartezeit angenehmer zu gestalten, indem ihm wechselnde Aphorismen und Lebensweisheiten präsentiert werden.

Die Menge und Ausdrucksstärke der angebotenen Optionen ist im Wesentlichen bei allen vorgestellten Meta-Suchmaschinen vergleichbar. Durch diese Optionen wird sowohl eine gewisse Individualität als auch eine Bandbreite bei den Benutzern unterstützt, wenn auch nicht in besonderem Maße.

A5 (Vollständige Anfrageplanung und -bearbeitung): Eine vollständige Anfrageplanung und -bearbeitung ist mit den vorgestellten Meta-Suchmaschinen nicht möglich. Das betrifft zum einen die eingebundenen Suchmaschinen. Beispielsweise enthält keine der beschriebenen Meta-Suchmaschinen Google, Northern Light oder HotBot, welche zweifelsfrei zu den größten und bekanntesten Suchmaschinen gehören. Lediglich ProFusion integriert Altavista und Yahoo. In der Regel werden nur wenige große und viele kleinere Suchmaschinen eingebunden. Die Vollständigkeit ist allerdings auch nicht gewährleistet, was die präsentierte Ergebnismenge betrifft. Meist werden von den eingebundenen Suchmaschinen nur die ersten Ergebnisse berücksichtigt, die in einem gewissen Zeitrahmen beschafft werden konnten. Der Benutzer kann diese Obergrenzen nur in einem vorgegebenen Rahmen variieren. Bei den vorgestellten Meta-Suchmaschinen schneidet MetaGer diesbezüglich am besten ab, hier besteht so gut wie keine Beschränkung in der Trefferanzahl und die Zeitobergrenze von 200 Sekunden ist deutlich höher als bei den anderen Meta-Suchmaschinen.

A6 (Duplikatbehandlung): Duplikate werden von allen vorgestellten Meta-Suchmaschinen erkannt und auch sofort eliminiert. Zu jedem Duplikat wird angegeben, von welchen Suchdiensten es gefunden wurde. MetaGer und ProFusion bereinigen die Ergebnismenge sogar noch zusätzlich, so dass dem Benutzer nur aktuelle und verfügbare Treffer präsentiert werden.

A7 (Schnelle Antwortzeiten): Den schnellen Antwortzeiten wird von jeder vorgestellten Meta-Suchmaschine eine große Bedeutung zugemessen, das zeigen beispielweise die Standardeinstellungen bei den Suchoptionen. Hierbei werden jeweils nur kleine Ergebnismengen von den schnellsten Suchdiensten angefordert.

A8 (Berücksichtigung von Lastvorgaben): Die Berücksichtigung von Lastvorgaben wird ebenfalls durch die vorgegebenen Standardeinstellungen gesichert, da bekanntlich die meisten Benutzer diese Einstellungen übernehmen (vgl. 3.2.2.2). Die Benutzer können zwar auch mehr Ergebnisse von den einzelnen Suchmaschinen anfordern oder eine längere Wartezeit angeben, allerdings nur in einem begrenzten Rahmen. Vivisimo lässt 500 Treffer in 30 Sekunden zu, ProFusion und MetaCrawler erlauben 30 Treffer pro Suchdienst in 2 Minuten. MetaGer begrenzt die Suchzeit auf 200 Sekunden. Ob diese Beschränkungen der Suche den Lastvorgaben der zugrundeliegenden Suchdienste Rechnung tragen oder vielmehr der Ungeduld des Benutzers, ist so einfach nicht zu beurteilen. Eine vollständige Meta-Suche im WWW ist allerdings auch nicht zwingend erforderlich, da die Informationen im Web per se weder vollständig sind noch einer Qualitätssicherung unterliegen.

3.3.2 Meta-Recherchedienste für die Literatursuche

In diesem Abschnitt betrachten wir existierende Meta-Recherchesysteme, welche Kataloge zur wissenschaftlichen Literatursuche integrieren. Dabei berücksichtigen wir nur Systeme, die technisch gesehen auch einen echten Meta-Rechercheansatz darstellen. Im Bibliotheksbereich findet man viele Zusammenschlüsse von Katalogen oder Dokumentbeständen, beispielsweise Bibliotheksverbünde (SWB, GBV), die Informatik-Bibliographie (CSBIB), DBLP, NCSTRL oder MeDoc. Bei diesen Ansätzen werden entweder die Dokumentbestände redundant in verschiedenen Einzelsystemen eingespielt (Bibliotheksverbünde), die Dokumentbestände im Wesentlichen manuell erweitert und gepflegt (DBLP, CSBIB) oder es wird eine einheitliche verteilte Technik verwendet (NCSTRL, MeDoc), in der sich jeder beteiligte Katalog an vorgegebene Protokolle und Schnittstellen halten muss.

Im Folgenden wollen wir nun drei bekannte Meta-Recherchedienste (jeweils Stand Dezember 2002) für den Bereich der Literatur ausführlicher vorstellen.

KVK. Der Karlsruher Virtuelle Katalog (KVK) [Mön01] ist das bekannteste und auch erfolgreichste Beispiel für ein echtes Meta-Recherchesystem im Literaturbereich. Der KVK wurde 1996 an der Universitätsbibliothek in Karlsruhe entwickelt und integriert zur Zeit mehr als 25 verschiedene Einzeldienste über ihre Webschnittstelle, darunter die größten nationalen und internationalen Bibliotheken sowie Bibliotheksverbünde und Vertreter des Online-Buchhandels. Täglich erhält der KVK mehr als 50 000 Benutzeranfragen aus aller Welt. Im Jahr 1999 erhielt der KVK den INETBIB Award. Realisiert ist der KVK in der Skriptsprache TCL, die Anwendung wird über den CGI-Mechanismus aufgerufen.

Zur Formulierung einer Anfrage stehen dem Benutzer zahlreiche Suchfelder wie Titel, Autor, Jahr, Verlag, ISBN oder Schlagwort zur Verfügung. Die gefundenen Dokumentennachweise werden dem Benutzer nur auf Kurztitelebene in einheitlicher Form präsentiert. Von jedem Dienst wird nur eine beschränkte Menge von Dokumentennachweisen angezeigt, nämlich die Menge, die der Dienst auf seiner ersten Ergebnisseite aufführt. Die Ergebnismengen der einzelnen befragten Dienste werden nicht miteinander vermischt. Die Reihenfolge der Ergebnisportionen wird durch die Antwortzeit der Dienste bestimmt, d.h. die Dokumentennachweise von schneller antwortenden Diensten erscheinen weiter oben in der Trefferliste. Für jede Ergebnisportion bzw. für jeden befragten Dienst wird seit September 2002 auch explizit die benötigte Antwortzeit aufgeführt.

Will der Benutzer nun mehr Informationen zu den präsentierten Dokumentennachweisen oder mehr Dokumentreferenzen der einzelnen Dienste sehen, muss er selbst manuell die angegebenen Links verfolgen. Damit verlässt er den einheitlichen Bereich des KVK und gelangt auf die Webseiten der zugrundeliegenden Einzeldienste. Dazu kann der Benutzer schon bei seiner Anfrage spezifizieren, ob er diese zusätzlichen Informationen im selben oder in einem separaten Fenster anschauen möchte. Wählt er die Anzeigeform mit einem separaten Fenster, bleibt ihm der sichtbare Bezug zur KVK-Recherche erhalten.

Zur Kontrolle über die vom KVK auf einzelne Dienste ausgeübte Last wird nur eine bestimmte Anzahl gleichzeitiger Anfragen verschiedener KVK-Benutzer an diese Dienste zugelassen. Für die Benutzung dieser Dienste aus dem KVK müssen Cookies und JavaScript im Browser des Benutzers aktiviert sein.

Eine Duplikaterkennung wird vom KVK insofern unterstützt, als zusätzlich zu den präsentierten Ergebnisgruppen eine sortierte Liste der aufgeführten Kurztitel erzeugt werden kann. Auch diese Ergebnisportion wird mit der benötigten Zeit versehen und erscheint schließlich am Ende der Gesamtergebnisliste. Als weitere Option bei der Anfrageformulierung kann der Benutzer eine Zeitschranke angeben (voreingestellt sind 60 Sekunden). Alle Dienste, die innerhalb von dieser Zeit Ergebnisse liefern, werden im Gesamtergebnis des KVK berücksichtigt.

Der KVK ist in erster Linie ein Werkzeug zum umfassenden Recherchieren von Literatur. Die Beschaffung eines Dokuments aus einer Bibliothek, die für den Benutzer nicht vor Ort ist und für die er keine Nutzungsberechtigung hat, gestaltet sich eher schwierig. Falls die Bibliothek einem entsprechenden Fernleihverbund angehört, kann der Benutzer das Dokument in einigen Fällen über eine Fernleihe beziehen. Für die Karlsruher Benutzer vor Ort existiert eine Variante des KVK, der KGK (Karlsruher Gesamtkatalog), der nur Bibliotheken aus dem Karlsruher Raum integriert. Dadurch können die Benutzer die gefundenen Dokumente bei Interesse meist schneller und müheloser beschaffen.

Daffodil. Daffodil [FGK00] ist ebenfalls ein Meta-Recherchesystem für Literatur, allerdings ist es in erster Line als Forschungsprototyp entwickelt worden und wird noch nicht für den operativen Betrieb verwendet. Daffodil ist auf den Bereich der Informatikliteratur beschränkt und hat sich dabei weiter auf Forschungsartikel spezialisiert, da diese mittlerweile relativ häufig im elektronischen Volltext gefunden werden können. Daffodil integriert etwa zwölf Dienste über ihre allgemein zugängliche Webschnittstelle. Daffodil ist als Java-Anwendung realisiert. Der Benutzer kann über die Java-Web-Start-Technik immer die aktuellste Klientensoftware von Daffodil über seinen Browser benutzen.

Daffodil versteht sich als Werkzeugkasten, der den gesamten Prozess der Literaturrecherche unterstützen will und ist daher nicht nur auf das Stellen von Anfragen und Begutachten von Ergebnissen ausgerichtet, sondern auf die Bereitstellung von einer Reihe von Werkzeugen, die dem Benutzer im Rahmen seiner Literaturrecherche weiterhelfen können. So gibt es neben der Rechercheschnittstelle einen Navigationsdienst für die ACM-Klassifikation, einen Generator von Autorennetzwerken, der auf Basis von gemeinsam publizierten Werken verwandte Autoren identifiziert, einen Thesaurus (http://www.cogsci.princeton.edu/~wn/, WordNet) als Formulierungshilfe für geeignete Anfrageterme, einen Navigationsdienst für das Verfolgen von Zitiert-Beziehungen zwischen Artikeln sowie einen Navigationsdienst für die Inhaltsverzeichnisse von Zeitschriften und Konferenzbänden. Die einzelnen Dienste bzw. Werkzeuge können dadurch angesprochen werden, dass der Benutzer eine Dokumenten-

referenz auf das entsprechende Dienstsymbol zieht (Drag&Drop). Personalisierung ist in Daffodil ebenfalls vorgesehen, d.h. der Benutzer kann für ihn interessante Dokumentennachweise oder Anfragen dauerhaft abspeichern und auch anderen Benutzergruppen bei Bedarf zugänglich machen. Auf diesen persönlichen Objekten kann der Benutzer Benachrichtigungsdienste aktivieren, so dass er über Änderungen, beispielsweise über jüngste Zitierungen eines Dokuments oder neue Anfrageergebnisse informiert wird.

Die Anfrageschnittstelle von Daffodil bietet nur einige Suchfelder an (Titel, Autor, Jahr und Schlagwort). Als Sprache der Dokumente kann man zwischen Deutsch und Englisch wählen. Zusätzlich kann der Benutzer den gewünschten Publikationstyp angeben, nämlich Dokumente, Bücher, Artikel, Konferenzbände oder Zeitschriften. Duplikate werden immer erkannt. Der Benutzer hat auf diesen Vorgang keinen Einfluss. Zur Bestimmung der Duplikate wird nach Eliminierung aller Sonderzeichen ein Vergleich der Titel vorgenommen.

Der Benutzer kann die Wartezeit für die Ergebnislieferung bestimmen, Standardwert sind 180 Sekunden, maximal möglich sind 450 Sekunden. Diese Option wurde nachträglich hinzugefügt aufgrund von ersten Benutzerstudien, in welchen die Benutzer hinsichtlich der langen Antwortzeiten irritiert waren [FKSM02]. Die Ergebnisse werden bei Daffodil nicht kontinuierlich geliefert, sondern erst dann, wenn sie vollständig vorhanden sind oder die angegebene Zeitgrenze erreicht wurde. In der Ergebnismenge werden nur die Kurztitel präsentiert. Will der Benutzer mehr Informationen über die Dokumente erfahren, muss er auch hier einen zusätzlichen Verweis verfolgen. Allerdings führt dieser Verweis nicht direkt zu den zugrundeliegenden Diensten. Die Zusatzinformationen werden dem Benutzer im einheitlichen Ergebnisformat von Daffodil präsentiert. Die Einzeldienste werden dazu für den Benutzer transparent befragt. Allein an der langen Wartezeit für die Beschaffung dieser Zusatzinformationen machen sich die Zusatzanfragen bzw. die Aufrufe von den zugrundeliegenden Einzeldiensten bemerkbar.

Die Beschaffung eines Dokuments im Volltext ist mit Daffodil immer dann möglich, wenn einer der befragten Dienste das Dokument ebenfalls im Volltext anbietet.

AllBookStores. AllBookStores ist ein System für Preisvergleiche zwischen zahlreichen US-Buchhändlern. AllBookStores hat Zugriff auf etwa 30 Online-Kataloge von Buchhändlern. In Deutschland gibt es kein vergleichbares System, da hier nach wie vor die Buchpreisbindung gilt. Bisher war die Buchpreisbindung in Deutschland für ca. 90 % der Bücher vertraglich geregelt. Diese Regelung wurde nun noch weiter ausgedehnt und verschärft. Seit dem 1. Oktober 2002 ist das Buchpreisbindungsgesetz (BuchPrG) in Deutschland in Kraft getreten (vgl. http://www.bundesregierung.de/Themen-A-Z/Kultur-,9563/Buchpreisbindung.htm).

Bei AllBookStores sucht der Benutzer zunächst in einer Datenbank nach Büchern und zu einem gefundenen Buch kann er dann einen umfassenden Preisvergleich durchführen lassen. Schon bei der Suche wird zu jedem Buch sowohl der Listenpreis angegeben als auch der

billigste in letzter Zeit gefundene Preis (Angabe mit Datum), allerdings beinhaltet dieser Preis noch keine Liefergebühren.

Der Schwerpunkt von AllBookStores liegt auf der Beschaffung von Dokumenten, genauer gesagt dem Preisvergleich. Der Vorgang der Recherche wird nicht speziell unterstützt, zumal auch keine Aussagen darüber gemacht werden, in welcher Dokumentbasis der Benutzer zu Beginn recherchiert. Die graphische Gestaltung der Seiten lässt vermuten, dass AMAZON dahinter steckt, allerdings werden auch Bücher in der Datenbasis geführt, die von AMAZON nicht angeboten werden bzw. die sogar von gar keinem Buchhändler (zur Zeit) angeboten werden. Möglicherweise wurden die Datenbestände von einigen Buchhändlern a priori – wie bei einem Data Warehouse Ansatz – zusammengeführt und werden nun regelmäßig aktualisiert.

Die Ergebnisdarstellung ist sehr übersichtlich und ansprechend gestaltet, sie präsentiert dem Benutzer die verschiedenen Beschaffungsalternativen in tabellarischer Form, jeweils aufgeschlüsselt nach dem Buchhändler, den Buchkosten, der Lieferzeit, den Lieferkosten und den Gesamtkosten. Die Alternativen sind aufsteigend nach den Gesamtkosten sortiert.

Tabelle 3.6 ergibt sich, wenn man die drei Meta-Recherchesysteme für Literatur anhand der zuvor formulierten Kriterien für Meta-Suchmaschinen bewertet.

Meta-Recherchesystem	Parallele Suche	Ergebnis-integration	Duplikat-eliminierung	AND/OR		Trans-parenz	Vollst. Suche
KVK	v	–	– (sortierte Liste)	v	v	–	– max. 5 Minuten
Daffodil	v	gering	v (Titelgleichheit)	v	v	v	– max. 7,5 Minuten
AllBookStores	nur für Preise eines Doks	nur für Preise eines Doks	– (nicht notwendig)	v	v	v	nur für Preise eines Doks
v vorhanden; – nicht vorhanden							

Tabelle 3.6: Übersicht über die Funktionalität bekannter Meta-Recherchesysteme

Auffallend ist der geringe Integrationsgrad aller drei Meta-Recherchesysteme. Beim KVK bleibt nahezu die gesamte Integrationsarbeit dem Benutzer überlassen. Er muss sich über Verweise zusätzliche Treffer bzw. zusätzliche Informationen besorgen. Diese Verweise werden zwar direkt angeboten, müssen allerdings vom Benutzer manuell verfolgt werden.

Dabei gelangt er auf die Webseiten der Einzeldienste. Bei Daffodil ist ebenso ein explizites Verfolgen von Verweisen notwendig, allerdings bleibt dabei die Systemumgebung und das Layout einheitlich, so dass dadurch die Transparenz gewährleistet wird. Bei AllBookStores ist zumindest zu einem gegebenen Dokument die Preis- und Angebotsübersicht der Alternativen vollständig integriert und übersichtlich gestaltet.

Ausschlaggebend für den geringen Integrationsgrad der Meta-Recherchesysteme ist die Informationsstruktur, die die zugrundeliegenden Literaturdienste aufweisen. Bei der Ergebnisanzeige wird zunächst eine bestimmte Anzahl von Treffern mit Angaben zu Titel, Autor und Jahreszahl geliefert. Für weitere Treffer oder für zusätzliche Informationen zu einem bestimmten Treffer, wie beispielsweise für Angaben zu Verlag, ISBN oder Seitenanzahl, muss eine weitere HTML-Seite angefordert werden.

Definition 3.1: Werden die Ergebnisse einer Anfrage nicht auf einer HTML-Seite angezeigt, sondern müssen durch das zusätzliche Verfolgen von Verweisen erst komplettiert werden, sprechen wir von einer *Informationsportionierung* des Recherchedienstes.

Eine Duplikaterkennung wird beim KVK nur insofern angeboten, als die Benutzer zusätzlich eine sortierte Liste der Kurztitel erhalten können. Eliminiert bzw. zusammengeführt werden die Duplikate beim KVK nicht. Daffodil hingegen nimmt stets eine Duplikateliminierung vor, wenn auch in sehr einfacher Form, d.h. indem die Titel der Dokumente miteinander verglichen werden. AllBookStores liegen für die Bestimmung der Duplikate sämtliche bibliographischen Informationen des Ausgangsdokuments vor. Beispielsweise kann die Suche nach dem gleichen Dokument bei den angegliederten Suchdiensten über die ISBN erfolgen. Bei den Beschaffungsalternativen ist jedoch eine vollständige Aufzählung erwünscht, hier sollen keine Duplikate eliminiert werden.

Die Vollständigkeit einer Recherche ist beim KVK und bei Daffodil durch eine vorgegebene Zeitschranke begrenzt. Da allerdings nur die Suchergebnisse auf Kurztitelebene abgefragt werden, sind beide Zeitschrankenobergrenzen als ausreichend zu bewerten. Sie beziehen sich in erster Linie auf die initiale Antwortzeit der eingebundenen Dienste und sind auch noch für einige weitere Anforderungen von Kurztitelportionen ausreichend. Diese Form der Vollständigkeit, dass zumindest alle gefundenen Kurztitel eines Dienstes integriert werden, ist nur bei Daffodil möglich. Für eine Anfragebearbeitung, die auch die bibliographischen Informationen der zweiten Ebene mit einbezieht, wären die vorgegebenen Zeitschranken vermutlich zu restriktiv.

Bewertung

A1 (Einheitliche, integrierte Benutzungsschnittstelle): Der KVK schneidet sowohl bei der Bewertung der Einheitlichkeit als auch bei der Bewertung des Integrationsgrads am schlechtesten ab. Sobald der Benutzer Verweise der Kurztitelliste verfolgt, befindet er sich auf Webseiten von anderen Diensten. Die Ergebnisse der zugrundeliegenden Dienste werden

nur in minimaler Hinsicht integriert, d.h. es wird nur der erste Teil ihrer Kurztitelliste berücksichtigt. Daffodil integriert zwar auch keine bibliographischen Zusatzinformationen in das Gesamtergebnis, präsentiert die Angaben aber zumindest in einem einheitlichen Format. Die Benutzungsschnittstelle von AllBookStores ist für die Preisgegenüberstellung zu einem gegebenen Buch sowohl einheitlich als auch integriert. Die Rechercheschnittstelle ist ebenfalls einheitlich und integriert, setzt allerdings auf einem zentralen Datenbestand auf. Im allgemeinen lässt sich sagen, dass der Integrationsgrad der vorgestellten Systeme nicht hoch ist, da der Benutzer immer noch selbst sehr viele Verweise per Hand verfolgen und auswerten muss.

A2 (Einfache, intuitive Benutzerinteraktion): Eine einfache und intuitive Benutzerinteraktion ist bei allen vorgestellten Systemen gegeben. Der Benutzer findet beim KVK und bei AllBookStores technisch gesehen vertraute HTML-Formulare und anwendungsspezifisch gesehen die üblichen Anfragefelder für eine Literaturrecherche vor. Die hohen Benutzerzahlen beider Systeme unterstreichen das zusätzlich. Durch den Web-Start-Mechanismus ist auch Daffodil einfach zu benutzen. Die Integration der verschiedenartigen Werkzeuge durch den Drag&Drop-Ansatz erlaubt ebenfalls eine intuitive und komfortable Benutzung. Eine bekräftigende Benutzerstudie dazu liegt allerdings bislang nicht vor.

A3 (Ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit): Die Benutzungsschnittstellen der vorgestellten Systeme bieten die klassischen Suchfelder zum Formulieren einer Literaturrecherche an. Daffodil unterstützt zusätzlich noch eine Einschränkung der Suche auf bestimmte Publikationstypen. Beim KVK und bei Daffodil hat der Benutzer jeweils die volle Kontrolle über die Zusammenstellung der zu befragenden Literaturkataloge. Darüber hinaus kann der Benutzer in einem gewissen Rahmen die Zeitdauer für die Ergebnislieferung beeinflussen. AllBookStores führt dagegen automatisch einen Preisvergleich auf allen eingebundenen Diensten durch. Daffodil bietet als einziges System verschiedene Sortierungen für die Darstellung der Ergebnismenge an.

A4 (Berücksichtigung menschlicher Eigenschaften): Der KVK und AllBookStores stellen weder Erinnerungshilfen zur Unterstützung des Suchprozesses bereit noch unterstützen sie den Benutzer in seinen individuellen Vorlieben. Sie setzen beim Benutzer kein besonderes technisches Geschick und auch keine speziellen Fertigkeiten bei der Literaturrecherche voraus. Damit unterstützen sie die breite Masse der Benutzer recht gut, was sich auch in der regen Benutzung der Dienste widerspiegelt. Besonders viel Wert legen beide Systeme auf schnelle Antwortzeiten und tragen damit also der bekannten Ungeduld des Benutzers Rechnung. Daffodil hingegen zielt auf die Unterstützung des Informationssuchprozesses ab. Durch die Ansätze zur Personalisierung wird sowohl das Gedächtnis als auch die Individualität der Benutzer unterstützt. Für das Verständnis und den Umgang mit den angebotenen Werkzeugen ist allerdings eine gewisse Expertise im Bereich der Literaturrecherche vonnöten. Der gravierendste Nachteil liegt jedoch in den langen Wartezeiten beim Recherchieren.

A5 (Vollständige Anfrageplanung und -bearbeitung): Eine vollständige Anfrageplanung und -bearbeitung ist mit keinem der vorgestellten Systeme möglich. Der KVK bietet diesbezüglich weniger Unterstützung als Daffodil. Daffodil ist zumindest in der Lage, die gesamte Menge von Kurztiteln von einem Dienst zu beziehen. AllBookStores ist im Grunde kein System, welches eine Meta-Recherche anbietet, in dem angestrebten Bereich der Preisvergleiche arbeitet AllBookStores allerdings vollständig.

A6 (Duplikatbehandlung): Die Duplikatbehandlung wird im KVK nur durch die zusätzliche Sortierung der Kurztitel angedeutet. Bei Daffodil werden Duplikate stets eliminiert, wenn auch mit einem sehr einfachen Verfahren. Tolerantere oder gar anpassbare Algorithmen werden zur Duplikatbehandlung in beiden Systemen nicht vorgesehen. Bei AllBookStores ist eine Duplikatbehandlung nicht notwendig.

A7 (Schnelle Antwortzeiten): Die Forderung nach schnellen Antwortzeiten werden vom KVK und von AllBookStores zweifelsfrei erfüllt. In der aktuellen Version vom KVK werden mittlerweile auch die Antwortzeiten der einzelnen Dienste erfasst und dem Benutzer mitgeteilt. Somit sieht der Benutzer, dass der KVK die Ergebnisse nicht schneller liefern kann als die zugrundeliegenden Dienste selbst. Um die Ergebnisse dem Benutzer so schnell wie möglich präsentieren zu können, erfolgt die Ergebnisanzeige inkrementell, d.h. sobald die Kurztitel eingetroffen sind. Dadurch fallen ausstehende Ergebnisse von langsameren Diensten nicht so sehr ins Gewicht, da der Benutzer bereits mit der Betrachtung der schnell eingetroffenen Ergebnisse beginnen kann. Das ist auch der Grund, warum die Duplikaterkennung beim KVK nicht in die Ergebnisanzeige integriert ist. Die langen Antwortzeiten von Daffodil sind sicherlich auch darauf zurückzuführen, dass kein inkrementeller Ergebnismengenaufbau erfolgt. Die Ergebnisanzeige erscheint, nachdem die Duplikate vollständig ermittelt worden sind.

A8 (Berücksichtigung von Lastvorgaben): Eine Berücksichtigung von Lastvorgaben wird nur beim KVK deutlich. Obwohl das Systemdesign schon eine möglichst minimale Last auf den Einzeldiensten garantiert, sind noch zusätzliche Einschränkungen in der Benutzung notwendig. Ursache dafür sind die großen Benutzerzahlen vom KVK, welche sich in der erzeugten Last genauso auf die eingebundenen Einzelsysteme auswirken. Diese Einzelsysteme sind teilweise nicht für so viele Benutzer ausgelegt worden.

3.3.3 Bewertung

In diesem Abschnitt haben wir die klassischen Meta-Suchmaschinen für das WWW sowie Meta-Recherchesysteme für den Literaturbereich betrachtet. Keine der beiden Systemklassen und auch keines der vorgestellten Einzelsysteme erfüllt den Anforderungskatalog vollständig.

Während Meta-Suchmaschinen als einheitlich und integriert bewertet werden können, erweist sich der Integrationsgrad der Meta-Recherchedienste als recht gering. Die gelieferten Informationen werden hier eher bruchstückhaft zusammengetragen oder lediglich zu

einzelnen Aspekten, wie beispielsweise dem Preisvergleich, vertieft. Die meiste Integrationsarbeit bleibt jedoch dem Benutzer selbst überlassen. Sicherlich liegt der mangelnde Integrationsgrad in der Struktur der zugrundeliegenden Literaturkataloge begründet. Diese liefern die Informationen zu einem Dokument nämlich ebenfalls portionsweise. Für die direkte Nutzung durch den Benutzer sind diese Informationsportionen auch angemessen, für die Nutzung im Rahmen eines Meta-Recherchesystems sind sie allerdings sehr ungünstig.

Insofern wird deutlich, dass die im täglichen Betrieb eingesetzten Meta-Recherchesysteme – genauso wie die Meta-Suchmaschinen – ihr Augenmerk auf schnelle Antwortzeiten legen. Dadurch können auch gleichzeitig die Lastvorgaben von Seiten der eingebundenen Dienste erfüllt werden. Bei Meta-Suchmaschinen kann der Benutzer zumindest in Grenzen die Vollständigkeit einer Suche hinsichtlich Suchzeit oder geforderter Trefferanzahl beeinflussen, beim KVK werden dem Benutzer dazu keine Variationsmöglichkeiten angeboten. Bei Meta-Suchmaschinen für das WWW haben wir argumentiert, dass die Erwartungen an die Suchergebnisse nicht so hoch sind und existierende Systeme trotz mangelnder Vollständigkeit durchaus ausreichend schnell und komfortabel für den Benutzer sind. Bei der Literaturrecherche verhält es sich dagegen anders: An eine wissenschaftliche Literatursuche werden sehr hohe Ansprüche gestellt, Literaturlisten sollen vollständig sein und auch aktuellste Arbeiten berücksichtigen. Die Qualität des Datenmaterials in den Einzelkatalogen ist gesichert und so sollten auch dem Benutzer Möglichkeiten zur Verfügung gestellt werden, um eine vollständige Suche durchführen zu können.

Bei Meta-Suchmaschinen stehen dem Benutzer zahlreiche Steuer- und Kontrollmöglichkeiten für den Vorgang der Suche zur Verfügung. Um die Handhabung der vielen Optionen zu unterstützen, bieten Meta-Suchmaschinen die Speicherung von persönlichen Einstellungen an. Bei den Meta-Recherchesystemen werden deutlich weniger Optionen angeboten, sie geben dem Benutzer wesentlich weniger Raum für Kontrolle, Einflussnahme und Individualität. Der Forschungsprototyp Daffodil lässt zwar auch keine Kontrolle des Recherchevorgangs zu, erlaubt aber die Speicherung von persönlichen Anfragen, Optionen und Ergebnissen.

Alle vorgestellten Meta-Suchmaschinen führen standardmäßig eine Duplikaterkennung und -eliminierung durch. Allerdings ist die Überprüfung der URL-Gleichheit auch nicht so aufwendig wie die Bestimmung von Duplikaten im Bereich der Literatur. Hier sind nämlich flexiblere Verfahren notwendig. Die Titelgleichheit ist dazu nicht ausreichend und die ISBN-Gleichheit ist möglicherweise zu restriktiv, falls man aufeinanderfolgende Auflagen desselben Buches als identisch betrachten möchte. Der KVK bietet keine Duplikateliminierung an, Daffodil arbeitet mit Titelgleichheit, was zumindest im Bereich von wissenschaftlichen Publikationen (hauptsächlich Artikeln) vertretbar ist.

Bei der Ergebnispräsentation bieten Meta-Suchmaschinen verschiedene hilfreiche Sortieroptionen an, die dem Benutzer den Überblick und die Begutachtung der Ergebnisse

erleichtern. Sogar ungültige Verweise werden automatisch eliminiert. Meta-Recherchesysteme hingegen bieten diesbezüglich wenig Unterstützung an. Eine Literatursuche endet oft mit einer Beschaffung, d.h. der Entscheidung des Benutzers, ein Buch auszuleihen oder zu kaufen. Von den vorhandenen Meta-Recherchesystemen werden diese Entscheidungen nicht unterstützt. Teilweise wird eine derartige Entscheidung durch die Verwendung des KVK sogar noch erschwert, da dem Benutzer zahlreiche Bücher präsentiert werden, die er gar nicht oder nur auf Umwegen (über Fernleihe) beschaffen kann. Benutzerstudien haben gezeigt, dass Benutzer ihre Entscheidungen nicht nur hinsichtlich des Inhalts, sondern auch hinsichtlich Verfügbarkeit, Aktualität und anderen zusätzlichen Kriterien treffen. Daher sollten diese Informationen zu den Dokumenten auch bei der Ergebnispräsentation bereitgestellt werden, damit auf dieser Basis dann Sortieroptionen und andere Hilfsmittel zur Entscheidungsunterstützung des Benutzers eingesetzt werden können.

Der dringendste Handlungsbedarf besteht bei Meta-Recherchesystemen nun also darin, für den Benutzer eine aussagekräftige Informationsbasis bzw. Ergebnismenge zu schaffen und diese mit Hilfsmitteln zur Entscheidungsunterstützung zu versehen. Dabei dürfen natürlich die anderen Anforderungen, besonders die schnellen Antwortzeiten und die Berücksichtigung der Lastvorgaben nicht vernachlässigt werden.

3.4 I^3-Architekturen

In diesem Abschnitt werden Ansätze betrachtet, die ihren Schwerpunkt auf die technische Integration der Einzeldienste legen. Dabei handelt es sich meist um Forschungsprototypen, die durch aufwendigere Integrationsmechanismen mit einer größeren Heterogenität der zugrundeliegenden Dienste umgehen können. Ziel ist es bei diesen Ansätzen, dem Benutzer einen einzigen Zugangspunkt zur Verfügung zu stellen, über welchen er auf die zur Verfügung stehenden Informationen in einheitlicher Form zugreifen kann.

Derartige Ansätze werden in der Forschung als I^3-Architekturen bezeichnet, I^3 steht dabei für die *I*ntelligente *I*ntegration von *I*nformationsquellen. Unter den Begriff der Informations-integration fallen allerdings eine ganze Reihe von Ansätzen, Mediatorarchitekturen, Data Warehouses, verschiedenartigste Informationsportale im Web sowie Systeme für den elektronischen Handel im B2B-Bereich (vgl. [Wie96], [Win01], [Kha02] oder [FOSB02]).

An dieser Stelle wollen wir die Gruppe der I^3-Architekturen berücksichtigen, die unserem Problembereich am ähnlichsten ist. Wir konzentrieren uns auf virtuelle Integrations-architekturen, die auf autonomen Datenquellen arbeiten und diese parallel anfragen können. Die integrierten Datenquellen können unterschiedlich und vielfältig sein, das Spektrum erstreckt sich von Informationsquellen, von denen die Daten recht leicht bezogen werden können, bis hin zu Informationsdiensten, die nur über spezielle APIs angesprochen werden können. Die Datenquellen sollen jeweils über das WWW kontaktiert werden. Abbildung 3.2 gibt einen Überblick über die wesentlichen Komponenten einer virtuellen

Integrationsarchitektur[4]. Die Datenquellen werden über Wandler (*wrapper*) einheitlich zugreifbar gemacht. Der Mediator ist über die zur Verfügung stehenden Datenquellen informiert. Eine Benutzeranfrage wird von der Mediatorkomponente nun so umformuliert, dass sie von den vorhandenen Wandlern möglichst gut und schnell beantwortet werden kann.

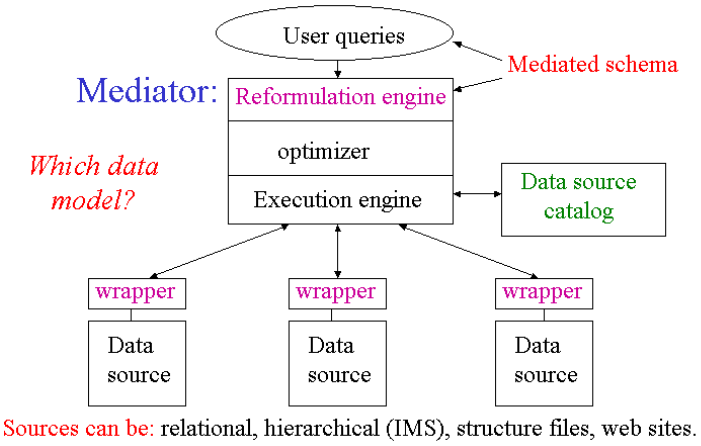


Abbildung 3.2: Wesentliche Komponenten einer virtuellen Integrationsarchitektur

Im Rahmen von Integrationsarchitekturen treten zahlreiche Herausforderungen auf. In Bezug auf die Heterogenität der eingebundenen Datenquellen sind die Anforderungen der vorliegenden Arbeit gering. Zum einen ist die Heterogenität der Dienste im Literaturbereich nicht so groß, zum anderen ist die vorliegende Arbeit Teil der UniCats-Architektur und kann somit auf die vorhandene Funktionalität der Wandler-Komponenten aufsetzen. Dasselbe gilt für die automatische Bestimmung der geeigneten Datenquellen zu einer gegebenen Benutzeranfrage. Sie kann für die vorliegende Arbeit ebenfalls vernachlässigt werden. Zum einen sollte der Benutzer bei einer Literaturrecherche die zu befragenden Dienste explizit angeben können, zum anderen kann im Rahmen der UniCats-Architektur auf die vorhandenen Trader-Komponenten zurückgegriffen werden, falls eine automatische Quellenauswahl gefordert wird.

Bei den folgenden Betrachtungen von verschiedenen Integrationsansätzen sollen die Schwerpunkte auf der Beschaffung und Kombination der verfügbaren Informationen liegen, auf der Aufbereitung der Informationen für den Benutzer sowie auf der Performanz des Gesamtsystems. Wir präsentieren die folgenden Ansätze in zwei Gruppen. In der ersten Gruppe stellen wir Integrationsansätze vor, die in den Jahren von 1995 bis 1999 entstanden sind, vorwiegend Informationsquellen integrieren und sich bei der Anfrageoptimierung auf die Erstellung von Anfrageplänen konzentrieren. Die zweite Gruppe der vorgestellten Ansätzen orientiert sich stärker an aktuellen Informationsdiensten im Web-Umfeld und

[4] Die Abbildung stammt aus Vorlesungsnotizen von Alon Halevy, s. http://rakaposhi.eas.asu.edu/cse494/notes/dataintegration-1.ppt.

berücksichtigt für die Anfrageoptimierung die Streaming-Eigenschaft und die portionsweise Lieferung der angebotenen Informationen. Oftmals sind die Ansätze der zweiten Gruppe Fortführungen der Projekte aus der ersten Gruppe.

3.4.1 Ansätze zur Integration von Informationsquellen

Information Manifold. Das Integrationsprojekt Information Manifold wurde bei AT&T unter der Leitung von Alon Levy (jetzt Halevy) durchgeführt. Dabei entstand ein System mit etwa 100 angeschlossenen Webquellen. Diese Quellen sind recht heterogen, sie enthalten Informationen zu Schauspielern und Filmen, zu diversen Fluglinien und rund um den Kauf von Gebrauchtwagen.

Information Manifold [LRO96b] verwendet zur Beschreibung und Vereinheitlichung der angeschlossenen Informationsquellen ein Domänenmodell (*world view*). Das Domänenmodell enthält eine Menge von Relationen, die die Informationen beschreiben, die in den angeschlossenen Quellen zu finden sind. Zwischen diesen Relationen können zusätzlich Klassenhierarchien mit Ober- und Untermengenbeziehungen und Vererbung angegeben werden. Die Relationen des Domänenmodells existieren lediglich virtuell. Sie dienen dem Benutzer als Vorlage, um seine Anfragen in einheitlicher Form (und auch deklarativ) spezifizieren zu können. Information Manifold unterstützt nur konjunktive Anfragen.

In Information Manifold werden die angeschlossenen Informationsquellen gemäß ihres Inhalts und ihrer Anfragemöglichkeiten modelliert. Der Inhalt einer Quelle wird als Anfrage an das Domänenmodell beschrieben. Eine Anfrage kann mehrere konjunktiv verknüpfte Bedingungen enthalten. Dadurch wird beschrieben, welchen Bedingungen (*constraints*) die Tupel der Quelle genügen. Implizit werden hier relationale Datenbasen als Quellen angenommen. Eine Quelle muss allerdings nicht unbedingt auch all die spezifizierten Datensätze enthalten. Quellen werden also als unvollständig angenommen. Zudem weisen die betrachteten Quellen Beschränkungen hinsichtlich der Anfragemöglichkeiten gegenüber voll funktionsfähigen Datenbanksystemen auf. Gerade wenn Datenbanken über Formulare im WWW zur Verfügung gestellt werden, weisen sie eine eingeschränkte Anfragefunktionalität auf. Daher werden zu einer Quelle explizit die Anfrage- und Ergebnisattribute modelliert sowie darauf aufsetzend die minimale bzw. maximale Anzahl von ausführbaren Selektionen. Diese vier Angaben werden als *query capabilitiy* einer Informationsquelle bezeichnet. Ziel bei der folgenden Anfragebearbeitung ist es, möglichst viel Funktionalität der Quellen auszunutzen, also die erforderlichen Selektionen möglichst von den Quellen selbst ausführen zu lassen.

Die Anfrageplanung bei Information Manifold erfolgt in zwei Schritten. Im ersten Schritt werden zu einer gegebenen Benutzeranfrage semantisch korrekte Pläne erzeugt, im zweiten Schritt werden diese Pläne auf ihre Korrektheit und Ausführbarkeit hin geprüft und ggf. umgeformt.

Für den ersten Schritt wird der *bucket*-Algorithmus vorgeschlagen und verwendet. Er berechnet für jedes Teilziel der Anfrage (für jede Konjunktion also), welche Informationsquellen zur Erfüllung des Teilziels relevant sind. Dabei ist eine Quelle relevant, falls die Informationsquelle ein Teilziel enthält, das mit dem gegebenen Teilziel unifiziert werden kann, so dass nach der Unifikation die Bedingungen der Anfrage und die Bedingungen in der Beschreibung der Informationsquelle wechselseitig erfüllbar sind. Weitere Details dazu findet man in [LRO96a]. Durch die recht fein granularen Quellenbeschreibungen wird dieser erste Schritt sehr gut unterstützt.

Im zweiten Schritt werden konjunktive Anfragepläne erzeugt, indem für jedes Teilziel der Benutzeranfrage eine relevante Informationsquelle aus dem zugeordneten Behälter (engl. bucket) ausgewählt wird. Es wird sozusagen ein Kreuzprodukt aller Behälter gebildet. Von allen resultierenden Plänen wird anschließend überprüft, ob sie relevant, korrekt und minimal sind. Minimal bedeutet an dieser Stelle, dass ein Plan eine minimale Anzahl von befragten Quellen enthält. Es darf also keine Quelle aus dem Plan weggelassen werden können. Alle produzierten Pläne sind korrekt, d.h. sie werden vorher so umgeschrieben, dass sie den Anfragefähigkeiten der benötigten Quellen entsprechen.

Der Aufwand für diese Überprüfung und Umschreibung ist exponentiell, und zwar in der Größe der Anfrage, nicht in der Anzahl der Quellen. Voraussetzung dafür ist die Vorgabe von maximal *einer* Anfrageeigenschaft (*query capability*) pro Quelle.

In [LMSS95] wurde gezeigt, dass das Problem des Findens von Anfrageplänen, die sich nur aus Anfragen an Sichten zusammensetzen, NP-vollständig in der Anzahl der vorhandenen Sichten ist, egal ob zusätzliche Bedingungen und Ordnungsprädikate bei den Sichten verwendet werden. Weiterhin wurde gezeigt, dass es genügt, Pläne mit einer Länge gemäß der Anzahl der Teilziele der Benutzeranfrage zu betrachten, um die Vollständigkeit der Beantwortung sicherzustellen. Diese Längenbeschränkung wird in Information Manifold zum Begrenzen (*pruning*) des Suchraums verwendet. Das Vorgehen ist allerdings nur anwendbar, wenn keine zusätzlichen Bedingungen oder Ordnungsprädikate bei den Sichten verwendet werden. In [KW96] wurde nämlich gezeigt, dass man mit zusätzlichen Bedingungen und Ordnungsprädikaten keine obere Schranke für die Länge eines Anfrageplans angeben kann.

In diesem Fall liefert also der in [LRO96b] vorgeschlagene Algorithmus keine vollständigen Informationen und müsste erweitert werden. Der Algorithmus findet also nicht alle Pläne, die für die Benutzeranfrage relevante Antworten liefern können.

Es wurden drei sehr einfache Anfragen mit bis zu 100 Quellen getestet. Die Pläne zu einer Anfrage konnten in weniger als 6 Sekunden generiert werden. Ein Plan konnte jeweils in weniger als einer Sekunde generiert werden. Die beschriebene Evaluierung bezieht sich lediglich auf den Schritt der Anfrageplanung. Über die Antwortzeiten der Anfrageausführung und Ergebnislieferung wurden keine Angaben gemacht.

SIMS/Ariadne: Das SIMS-System (*S*ervices and *I*nformation *M*anagement for decision *S*ystems) sowie das Nachfolgeprojekt Ariadne wurde von der Agenten-Forschergruppe der Südkalifornischen Universität unter der Führung von Craig A. Knoblock and Steven Minton durchgeführt [AK00]. Für beide Projekte sind mehrere Szenarien verfügbar: Die TheaterLoc-Anwendung findet das passende Restaurant im Anschluss an einen Kinobesuch. Es gibt Zusammenführungen von mehreren elektronischen Produktkatalogen sowie eine Anwendung für diverse Länderinformationen von der ganzen Welt.

Als Beschreibungsformalismus liegt SIMS ebenfalls ein Domänenmodell zugrunde. Dieses Modell wird mit der Wissensrepräsentationssprache LOOM, einem Mitglied der KL-ONE-Familie, beschrieben. Das Domänenmodell definiert ein festes Vokabular, welches Objekte der Domäne, ihre Attribute und Beziehungen zwischen ihnen beschreibt. Die Anfragen werden ebenfalls in dieser Sprache formuliert.

Darüber hinaus werden auch die Informationsquellen in LOOM modelliert. Von einer Quelle wird die Vollständigkeit der Informationen angenommen. Im Gegensatz zu Information Manifold wird hier bei der Quellenbeschreibung mit Gleichheit gearbeitet, bei Information Manifold hingegen wurde die Inklusion verwendet. Eine Ober- oder Untermengenbeziehung kann allerdings auch in SIMS modelliert werden. Für die Informationsquellen gelten keine Beschränkungen der Anfragemöglichkeiten. Im Wesentlichen geht man bei den Quellen von Relationen aus. Im Nachfolgeprojekt Ariadne wird der Schwerpunkt weg von relationalen Quellen, die mittels JDBC über das Internet angefragt werden können, hin zu allgemeinen Informationsquellen verlagert, die über die üblichen HTML-Formulare und CGI-Mechanismen angesprochen werden [KMAA01].

Ariadne baut auf der Beobachtung auf, dass die Erstellung der Anfragepläne zur Laufzeit sehr aufwendig ist und identische Teilpläne häufig mehrmals generiert werden. Die Grundidee ist daher, Teile von Anfrageplänen vorab zu erzeugen. Dies geschieht durch die Einführung sogenannter *integration axioms*, welche die Kombinationsmöglichkeiten von Quellen beschreiben. Die Quellenbeschreibungen werden in einem Vorverarbeitungsschritt für jede Klasse des Domänenmodells betrachtet und gemäß der maximal enthaltenen Menge von Attributen zusammengestellt.

Die Anfrageplanung basiert auf dem *Planning-by-Rewriting*-Paradigma (PbR). Zunächst wird eine Anfrage mithilfe der *integration axioms* so umformuliert, dass sie möglichst spezielle Klassen (also Unterklassen) der Informationsquellen enthält. Dann wird mit einer Tiefensuche ein erster Anfrageplan gesucht. Dieser ist suboptimal, kann dafür aber schnell gefunden werden. Der Anfrageplan wird dann iterativ im Hinblick auf Kostenoptimierung umgeformt. Die verwendeten Regeln berücksichtigen dabei die Kosten der relationalen Operatoren sowie Verteilungsaspekte. Prinzipiell können aber beliebige Regeln eingebracht werden. Die Suche zur Anwendung der nächsten Regel entspricht einem Gradientenabstiegsverfahren, welches ein lokales Minimum finden wird. Durch diese lokale Suche arbeitet der Planer sehr effizient

und skaliert gut. Im Gegensatz zu anderen Ansätzen können die Quellenauswahl und die Anfrageplanung miteinander verknüpft und im gleichen Schritt optimiert werden. Das liegt an der freien Gestaltung der verwendeten Regeln.

Bei den Operatoren zur Anfrageplanung wird keine Rekursion unterstützt. Gerade bei Webquellen ist im Vorhinein nicht bekannt, wie viele Ergebnisse eine Anfrage liefern wird. Meist müssen die Ergebnisse über eine „Weiter"-Funktionalität solange aufgesammelt werden, bis das Ende erreicht ist. Dieses Defizit wird in Ariadne dadurch umgangen, dass geschickt definierte Wandler diese Rekursionsfunktionalität erbringen, so dass sie auf Ebene der Anfrageplanung nicht mehr notwendig ist.

Um die Kosten der Anfragebearbeitung noch weiter zu optimieren, werden teure Teilanfragen materialisiert. Die Daten werden lokal abgelegt und als zusätzlich verfügbare Informationsquelle entsprechend modelliert. Diese Optimierung ist besonders für Webquellen geeignet, die nur begrenzte Anfragemöglichkeiten bereitstellen und für die die LOOM-Modellierung nicht angemessen ist. Beispielsweise hat das CIA World Fact Book (http://www.cia.gov/cia/publications/factbook/index.html) gar keine Anfragefunktionalität und die Informationen werden lediglich durch Navigation über die einzelnen HTML-Seiten gewonnen. Durch diese Materialisierungsstrategie können beliebige Webquellen unter Beibehaltung der LOOM-Modellierung integriert und in effizienter Weise in die Anfrageplanung mit einbezogen werden.

InfoMaster ist ein Projekt an der Universität Stanford, das unter die Federführung von Michael R. Genereth und Oliver Duschka [DG97b] durchgeführt wurde. Bei InfoMaster gibt es drei Abstraktionsebenen zur Darstellung der Informationen. Auf der obersten Ebene findet der Benutzer die *interface-relations*, die von ihm angefragt werden können. Darunter befinden sich die *base-relations*, die sozusagen als stabile Zwischenschicht angesehen werden können. Auf der untersten Ebene befinden sich die *site-relations*, die die Informationen beschreiben, die tatsächlich in den Quellen vorhanden sind. Sowohl die interface- als auch die site-relations werden als Sicht auf die base-relations definiert. Als Beschreibungsformalismus stehen Prädikate, positive boolesche Ausdrücke, nicht rekursive eindeutige Definitionen bestehend aus Kopf und Rumpf sowie Integritätsbedingungen zur Verfügung. Benutzeranfragen werden in positiver Relationaler Algebra formuliert.

Bei der Modellierung der Informationsquellen kann zwischen Enthaltensein (liberale Sicht) und Gleichheit (konservative Sicht) bezüglich der Basisrelationen unterschieden werden. Für die Anfrageoptimierung ist diese Unterscheidung wichtig, da dadurch ein erhebliches Maß an Redundanz gespart werden kann im Gegensatz zur Annahme einer offenen Welt, d.h. im Gegensatz zur Annahme, dass jede Informationsquelle immer nur eine Teilmenge bezüglich der Gesamtheit liefern kann.

Für die Anfragebearbeitung wird im ersten Schritt Qians Algorithmus [Qia96] verwendet, welcher semantisch korrekte und vollständige Anfragepläne erzeugt. Dieser wurde für

Anfragen in positiver Relationaler Algebra verallgemeinert, bisher setzten die vorgeschlagenen Algorithmen dagegen stets konjunktive Anfragen voraus. Im zweiten Schritt werden die redundanten Teile der erzeugten Anfragepläne eliminiert, so dass die Anfragepläne danach zusätzlich Sichten-minimal sind. Quians Algorithmus wurde zusätzlich noch dahingehend erweitert, dass er mit rekursiven Quellenbeschreibungen und Anfragen umgehen kann [DG97a].

Emerac wurde von Eric Lambrecht und Subbarao Kambhampati an der Arizona State Universität durchgeführt ([LK97], [LKG99]). Emerac verwendet ein *world model* als Domänenmodell, welches aus Relationen besteht. Als Anfragen werden nur konjunktive Anfragen zugelassen.

Bei der Modellierung der Informationsquellen werden ebenfalls Relationen verwendet, aber als Sichten auf die Relationen des Domänenmodells. Die Attribute, die als Eingabe bei einer Anfrage erforderlich sind, werden ebenso markiert wie die Attribute, auf denen keine Bedingungen angegeben werden können. In Emerac können Informationsquellen als vollständig oder unvollständig ausgezeichnet werden, aber auch als teilweise vollständig, über die Angabe von lokalen Annahmen der geschlossenen Welt (LCW statements). Dadurch wird der Bereich ausgedrückt, für den die Quelle als konzeptuell vollständig betrachtet werden kann. Weiterhin können Ober- und Untermengenbeziehungen modelliert werden, so dass dadurch sowohl die Unvollständigkeit also auch die Anfragemöglichkeiten präziser gehandhabt werden können.

Für die Anfrageplanung wird ein Satz von Regeln formuliert, mit welchen die Anfrage umformuliert werden kann. Diese Regeln werden zur effizienteren Verarbeitung invers abgelegt [DG97b]. Der Planungsalgorithmus wird über die LCW-Annahmen in Bezug auf Redundanz optimiert, allerdings wird hier nicht nur – wie bei den anderen vorgestellten Ansätzen – die Redundanz zweier Quellen berücksichtigt, sondern auch die Redundanz, dass eine Quelle durch die Kombination von mehreren anderen Quellen ersetzt werden kann. In Emerac können auch rekursive Anfragepläne gehandhabt werden [LKG99].

Bei diesem Ansatz werden zum ersten Mal End-zu-End-Antwortzeiten bei der Anfrageoptimierung betrachtet. Bei der Anwendung bzw. Minimierung der Regeln werden die über Heuristiken ermittelten Kosten einer Quellenbefragung berücksichtigt. Die Heuristiken entstehen aus Beobachtungen der vorangegangenen Anfragen. Prinzipiell können sich die Kosten jedoch beliebig zusammensetzen. Für Emerac wird als Kostenfunktion eine Kombination aus Kontaktzeit, Anfragezeit, Tupelantwortzeit, Zugriffswahrscheinlichkeit und Zugriffskosten favorisiert. Im allgemeinen kann man die Kosten einer Quellenbefragung einteilen in Zugriffskosten, Übertragungskosten und Verarbeitungskosten. Emerac setzt den Schwerpunkt hier auf die Zugriffskosten, insbesondere wird die Phase des Verbindungsaufbaus stark gewichtet.

3.4.2 Ansätze zur Integration von Web-basierten Informationsdiensten

Theseus ist das Nachfolgeprojekt von SIMS und Ariadne. Der Schwerpunkt liegt bei Theseus auf der flexiblen Erstellung von Anfrageplänen und auf Effizienz-Untersuchungen. Theseus verwendet zwei unterschiedliche Szenarien, die TheaterLoc-Anwendung und die Immobiliensuche nach einem neuen Eigenheim.

Theseus ist eine Ausführungsplattform für Informationsagenten [BK02a]. Das Ziel ist es, komplexe Anfragepläne leicht spezifizieren und optimiert ausführen zu können. Daher basiert Theseus auf einer Datenflussarchitektur, die sowohl horizontale (zwischen verschiedenen Operationen) als auch vertikale Parallelität (innerhalb einer Operation) zulässt.

Theseus gehört zur Klasse der *network query engines*. Diese Systeme befragen verschiedene Dienste im Web, wobei aus Ergebnissen durchaus auch wieder neue Anfragen gebildet werden können. Die Ergebnisse werden dann entsprechend transformiert, miteinander gemischt und gegebenenfalls gefiltert. Dem Benutzer sollen die Ergebnisse so schnell wie möglich präsentiert werden, sie können bei Bedarf aber auch asynchron via Mail verschickt werden.

Anfragepläne bestehen in der Regel aus den klassischen relationalen Operatoren wie Selektion, Projektion, Join, Vereinigung, Durchschnitt und Differenz. In Theseus enthält die Spezifikationssprache von Anfragepläne noch zusätzliche Operationen: *pack*, *unpack*, *dependent join* sowie die Spezifikation von Unterplänen und rekursiven Plänen. Für jeden Operator wird definiert, welche Hilfsdatenstrukturen er benötigt und wie schnell mit Ergebnissen zu rechnen ist. Ein Operator feuert, sobald er einen Datenstrom produziert und sofern der nachfolgende Operator zur Ausführung bereit ist. Durch Programmfäden (threads) wird die horizontale Parallelität erreicht, durch das Erzeuger-Verbraucher-Entwurfsmuster wird die vertikale Parallelität realisiert.

Aktuelle Arbeiten [BK02b] im Rahmen des Theseus-Projekts beschäftigen sich mit der spekulativen Ausführung von Operatoren bzw. Anfragen. Da einige Dienste lange Antwortzeiten benötigen, können aus vorangegangenen Anfragen und Ergebnissen Schätzungen abgeleitet werden. Damit kann die aktuelle Anfrage beispielsweise auch parallel an andere schnellere Dienste gestellt werden oder eine zu erwartende Anfrage schon vorab angestoßen werden. Natürlich muss anschließend auch immer die Korrektheit der Spekulation geprüft werden.

Eine einfache Benutzungsschnittstelle wurde für das Szenario der Immobiliensuche realisiert. Zur Zeit ist ein Benutzerassistent in Arbeit, welcher dem Benutzer einfache Fragen stellt und daraus dann die entsprechenden Anfragepläne zusammenbauen kann. Ergebnisse von Untersuchungen zur Effizienz wurden bisher noch nicht veröffentlicht.

Tukwila ist das Nachfolgeprojekt von Information Manifold. Der neue Schwerpunkt liegt bei Tukwila in einer flexibleren Anfrageoptimierung und -ausführung, die ohne Statistiken über die vorliegenden Daten auskommt, mit unvorhersehbaren und wechselhaften Übertragungs-

raten im Web umgehen kann und im Falle von nicht verfügbaren Datenquellen Alternativen finden kann [IFFL99].

Der Lösungsansatz von Tukwila besteht im Einsatz von adaptiven Operatoren. Diese können sich je nach vorliegenden Ergebnismengen, -größen, -geschwindigkeiten und -verteilungen anpassen. Somit ist die Anfrageplanung mit der Ausführung verzahnt, die Planung kann anfangs nur partiell erfolgen, dafür aber auch situationsgemäß nachgebessert werden.

Für die Anfragebearbeitung wird ein dynamischer Sammeloperator (*union*) definiert, welcher aufgrund von gegebenen Bedingungen die zu vereinigenden Datenquellen bestimmen kann. Hier liegt die Annahme zugrunde, dass der Grad der Überlappung von den Datenquellen jeweils bekannt ist und im Allgemeinen auch häufig eine Überlappung zwischen den Datenquellen auftritt. Als effiziente Implementierung des Join-Operators wird der *double pipelined hash join* vorgeschlagen. Auf zwei kontinuierlichen Datenströmen lassen sich nämlich weder der *sort-merge join* noch der *index join* noch der *nested loop join* und auch nicht der *hash join* effizient anwenden. Kritisch beim hash join ist beispielsweise, ob die innere Relation in den Hauptspeicher passt. Die Vorteile des double pipelined hash join bestehen darin, dass er ein symmetrischer Operator ist, dass die Zeit zum Eintreffen des ersten Tupels im Ergebnis minimiert wird und dass er datengesteuert ist, d.h. ein langsamer Datenstrom kann durch einen schnelleren Datenstrom kompensiert werden.

Im Tukwila-Projekt wurde darüber hinaus der *MiniCon*-Algorithmus entwickelt [PL00], der eine Verbesserung des bucket-Algorithmus (Information Manifold) und des reverse-rules-Algorithmus (InfoMaster) im Hinblick auf Skalierbarkeit darstellt. Ein weiterer Beitrag des Tukwila-Projekts ist der *x-scan*-Operator [ILWF00]: Auf XML-Dokumenten müssen häufig reguläre Pfadausdrücke ausgewertet werden. Die übliche Vorgehensweise besteht darin, dass das XML-Dokument intern als Menge von Tabellen oder Objekten abgelegt wird und dann entweder Joins oder zusätzliche Indexstrukturen für die Auswertung der Ausdrücke verwendet werden. Das ist gerade bei virtuellen Integrationsarchitekturen zu aufwendig, da die Zwischenergebnisse meist nicht gespeichert bzw. wiederverwendet werden. Der x-scan-Operator kann auf Strömen von XML-Dokumenten angewendet werden. Er baut beim Einlesen der Daten einige Hilfsstrukturen auf und kann dadurch eine Menge von regulären Pfadausdrücken auswerten.

Telegraph entstand in einem Projekt an der Uni Berkeley [SMFH01]. Telegraph ist ein adaptives Datenflusssystem, welches es erlaubt, auf beliebige Daten im WWW oder auch innerhalb von Diensten im WWW (*hidden web*) zuzugreifen, diese zu kombinieren und ggf. weiter zu analysieren. Als erster Prototyp wurde Telegraph FFF (*F*ederated *F*acts and *F*igures) Election 2000 entwickelt. Diese Anwendung stellt Informationen über die US-Präsidentschaftswahlen im Jahr 2000 zusammen, beispielweise die Kosten der verschiedenen Kampagnen, Informationen über Spender und Sponsoren, Informationen über Prominente, etc. Das System wurde in dieser Zeit von mehreren Tausend Benutzern verwendet.

Die klassischen relationalen Operatoren eines Anfrageplans können bei Telegraph in Reihen (*pipelines*), Bäumen oder Graphenstrukturen miteinander kombiniert werden. Dazu wurden zwei neue adaptive Techniken entwickelt: *Eddies* [HA00] bauen den Operator-Graphen kontinuierlich um, um damit die Performanz zu optimieren, beispielsweise je nach aktueller Datenrate, und erzeugen dadurch einen höheren Grad an vertikaler Parallelität. *Rivers* verteilen die entstehende Last auf mehrere Maschinen und erhöhen damit den Grad an horizontaler Parallelität.

Telegraph unterstützt alle Operatoren der Relationalen Algebra, bevorzugt zur Implementierung des Joins die Ripple-Join Familie. Diese Joins weisen viele symmetrische Momente auf, an denen eddies beispielsweise die Ausführungsreihenfolge an die aktuelle Datenrate anpassen können. Zur Voroptimierung werden Heuristiken eingesetzt, um die Join-Reihenfolge zu bestimmen. Die kleinen Relationen werden möglichst früh über das Kartesische Produkt verbunden, dann werden Equi-Joins ausgeführt, wobei stark selektive Joins bevorzugt werden, und schließlich folgen die übrigen Joins. Als Algorithmen werden, sofern Indexe vorhanden sind, der *index join*, ansonsten der *hash ripple join* und für ganz allgemeine Joins der *block ripple join* verwendet.

3.4.3 Bewertung

A1 (Einheitliche, integrierte Benutzungsschnittstelle): Die Einheitlichkeit der Benutzungsschnittstellen ist bei allen vorgestellten Ansätzen gegeben, die mit einem Domänenmodell arbeiten. Der Integrationsgrad kann nur anhand von existierenden Benutzungsschnittstellen beurteilt werden. Prototypen existieren lediglich von SIMS/Ariadne (TheaterLoc), Theseus (Immobiliensuche) und Telegraph (FFF Election 2000). Der Integrationsgrad von diesen Anwendungen kann als mittelmäßig bewertet werden. Der Benutzer muss trotz des einheitlichen Zugangspunkts zahlreiche Verweise manuell verfolgen bzw. Informationen explizit kombinieren. Da in jedem der Szenarien recht verschiedenartige Informationen miteinander verknüpft werden, können diese Kombinationsmöglichkeiten nicht alle automatisch im Voraus berechnet und vorgehalten werden.

A2 (Einfache, intuitive Benutzerinteraktion): Eine einfache und intuitive Benutzerinteraktion ist bei den vorhandenen drei Benutzungsschnittstellen insofern gegeben, als sie lediglich die für den Benutzer gewohnten Gestaltungsmittel und Interaktionsformen von Benutzungsschnittstellen im WWW verwenden. Da eine Immobiliensuche bei Theseus doch recht unterschiedlich verlaufen kann, wird zur Zeit ein Benutzerassistent entwickelt, welcher in einem geführten Dialog, d.h. durch eine Reihe von einfachen Fragen an den Benutzer, konkretere Anforderungen oder Vorstellungen über die gesuchte Immobilie und damit über den einzuschlagenden Suchverlauf herausfinden kann. Die anderen Ansätze geben die von ihnen unterstützten Anfragesprachen an, beispielsweise konjunktiv verknüpfte Prädikate oder Bedingungen bzw. Anfragen in positiver Relationaler Algebra. Ohne eine zusätzliche Unterstützung ist eine derartige Anfrageformulierung von Benutzern nicht zu erwarten. Bei

den meisten Ansätzen liegt der Schwerpunkt allerdings auch mehr auf den Herausforderungen der technischen Integration als auf der benutzungsgerechten Gestaltung des Systems.

A3 (Ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit): Eine ausdrucksstarke Benutzerinteraktion mit Kontroll- und Steuermöglichkeiten für die Anfrageausführung wird von den vorgestellten Ansätzen nicht unterstützt. Ausdrucksmächtig sind die Anfragesprachen zwar, allerdings sind sie gleichzeitig auch deskriptiv, so dass der Benutzer über die Anfrageformulierung keinen Einfluss auf die Ausführung nehmen kann.

A4 (Berücksichtigung menschlicher Eigenschaften): Die Berücksichtigung menschlicher Eigenschaften liegt bei allen vorgestellten Ansätzen nicht im Fokus der Arbeit. Allenfalls wird von Ariadne, Emerac und der zweiten Gruppe der Ansätze der Ungeduld des Benutzers Rechnung getragen, indem eine möglichst schnelle Beantwortung der Anfrage bzw. Lieferung der Ergebnisse angestrebt wird.

A5 (Vollständige Anfrageplanung und -bearbeitung): Eine vollständige Anfrageplanung wird von den meisten vorgestellten Ansätzen durchgeführt. Allerdings soll dabei gleichzeitig auch das Kriterium der Minimalität eines Anfrageplans erfüllt werden. Die beschriebenen Ansätze unterscheiden sich hinsichtlich der Annahme einer offenen bzw. einer geschlossenen Welt. Ansätze mit der Annahme einer offenen Welt (z.B. Information Manifold) erzeugen sozusagen so vollständige Ergebnisse wie möglich, da sie alle Quellen befragen, die für die Anfrage relevant sein können. Die Annahme der geschlossenen Welt ist hingegen für die Minimierung von Anfrageplänen hilfreich, da damit Redundanzen besser identifiziert und eliminiert werden können. Allerdings entspricht die Annahme der geschlossenen Welt nicht der Realität, die man momentan bei Informationsquellen im WWW antrifft. Nahezu kein Dienst liefert so lückenlose und umfassende Daten, als dass man sie nicht noch durch die Befragung eines anderen Dienstes ergänzen könnte.

A6 (Duplikatbehandlung): Die Duplikatbehandlung wird bei den vorgestellten Ansätzen nicht explizit erwähnt. Bei einigen Ansätzen ist es sogar das Ziel, aufgrund der geschlossenen-Welt-Annahme mögliche Redundanzen im Vorhinein auszuschließen. Ansonsten ist die Notwendigkeit einer Duplikatbehandlung eher vom Szenario bedingt bzw. eine Frage der Präsentation. Bei den drei realisierten Anwendungsbeispielen treten aufgrund der gewählten Quellen gar keine Duplikate in Erscheinung.

A7 (Schnelle Antwortzeiten): Die erste Gruppe der vorgestellten Ansätze richtet ihr Augenmerk hauptsächlich auf die Komplexität der Anfrageplangenerierung. Bei Information Manifold beispielsweise dauert die Anfrageplanung von einfachen Anfragen schon bis zu 6 Sekunden. Die Zeiten für die Anfrageausführung wurden dabei nicht berücksichtigt und für die Anfrageoptimierung wurde die Annahme getroffen, dass jeder Quellenzugriff gleich teuer ist. Die zweite Gruppe der Ansätze hingegen berücksichtigt die charakteristischen Eigenschaften von Diensten im WWW und sieht für schnelle Antwortzeiten Mechanismen für die nicht blockierende Bearbeitung von Datenströmen (z.B. double pipelined hash join, x-

scan, eddies), das Ersetzen von gerade nicht verfügbaren Quellen (Theseus, Tukwila) oder die spekulative Ausführung von langsamen Anfragen (Theseus) vor.

A8 (Berücksichtigung von Lastvorgaben): Bei den vorgestellten Ansätzen werden viele Eigenschaften von Informationsquellen betrachtet, wie beispielsweise die Anfragemöglichkeiten oder das Antwortzeitverhalten. Die erzeugte Last bei der Anfrageausführung wird allerdings nicht berücksichtigt. Da alle Ansätze Forschungsprototypen sind, entsteht dabei auch weit weniger Last als bei Systemen im operativen Betrieb.

Der Beitrag der ersten Gruppe der vorgestellten Ansätze besteht im Wesentlichen in der Entwicklung von Verfahren zur Berechnung von semantisch korrekten, vollständigen und minimalen Anfrageplänen. Die dazu betrachteten Quellen sind sehr heterogen in Bezug auf ihre Inhalte. Beispielsweise werden für das TheaterLoc-Szenario Informationsquellen mit aktuellen Kinoprogrammen, Kinos, elektronischen Stadtplänen und Restaurants benötigt. Bei der Erstellung eines Anfrageplans geht es nun hauptsächlich darum, in welcher Kombination und Reihenfolge die gegebenen Quellen befragt werden müssen, um die gewünschten Informationen zu erhalten. Dazu müssen von jeder Quelle die geforderten Anfrageattribute und die gelieferten Ergebnisattribute beachtet werden.

Beim Literaturszenario, welches dieser Arbeit zugrunde liegt, handelt es sich dagegen um ein homogenes Szenario, sozusagen ein One-World-Szenario, in dem alle verfügbaren Dienste ausschließlich Informationen über Dokumente bereitstellen. Die Erstellung von Anfrageplänen wird also in erster Linie keine Betrachtungen der Kombinationsmöglichkeiten oder Reihefolgenabhängigkeiten benötigen. Die gewählten Quellen sollen in der Regel gleichzeitig parallel befragt werden. Dabei kann entweder der Benutzer die zu befragenden Quellen explizit angeben oder der Trader kann aufgrund von Angaben zum Fachbereich, dem gewünschten Publikationstyp oder bestimmten Beschaffungsbedingungen die geeigneten Quellen bestimmen. Die Anfragepläne sollen auch nicht minimal werden, da gerade verschiedene Zusatzinformationen oder Beschaffungsoptionen zum selben Dokument den Mehrwert des Systems ausmachen.

Die zweite Gruppe der vorgestellten Ansätze beschäftigt sich mit Anforderungen, die für die vorliegende Arbeit relevanter sind. Diese Ansätze entwickeln nicht blockierende Join-Algorithmen und alternative Ausführungspläne für langsame oder gerade nicht verfügbare Dienste im WWW. Im Literaturszenario werden die verschiedenen Dienste bzw. Dokumente allerdings nicht mit Joins verknüpft, sondern über eine Vereinigung. Die Idee der nicht blockierenden Operatoren kann jedoch bei der Optimierung von Antwortzeiten von großer Bedeutung sein. Daher sollte im weiteren Verlauf der Arbeit geprüft werden, wie sich diese Vorgehensweise beim Umgang mit der Menge von Informationsportionen und besonders beim Erkennen und Eliminieren von Duplikaten umsetzen lässt.

3.5 Benutzerunterstützung

In diesem Abschnitt betrachten wir Ansätze zur Unterstützung des Benutzers bei der Interaktion mit einem Recherchesystem. Dabei sind zwei Phasen der Recherche von großer Bedeutung, die Formulierung der Anfrage und die Präsentation der Ergebnismenge. Zur Unterstützung bei der Anfrageformulierung betrachten wir Personalisierungstechniken, die zudem den Vorteil haben, dass sie die vorhandene Informationsflut individuell auf den Benutzer zuschneiden und damit besser handhabbar für ihn machen. Anschließend betrachten wir Ansätze zur graphischen Präsentation der Ergebnisse. Diese Ansätze argumentieren, dass der Mensch durch geeignete graphische Darstellungsformen die Information besser wahrnehmen und begreifen kann als in textueller Form.

3.5.1 Personalisierung

Unter Personalisierung versteht man, dass unterschiedliche Benutzer eine Menge von Informationen auf unterschiedliche Weise präsentiert bekommen, d.h. sie sehen unterschiedliche Dinge, entweder andere Teilbereiche der Information oder anders aufbereitete und dargestellte Inhalte. Personalisierung ist zu einem Schlagwort geworden, das in verschiedenen Zusammenhängen und Anwendungsszenarien an Bedeutung gewonnen hat. Oft wird es von Firmen allerdings benutzt, um ihr Customer Relationship Management (CRM) geschickter und mehr von der technischen Seite her zu verkaufen. Wir wollen in diesem Abschnitt Ansätze betrachten, die den Benutzer bei der Informationssuche individuell unterstützen, persönlich zugeschnittene Empfehlungen liefern (*recommender systems*) oder Informationen nach persönlichen Vorlieben filtern (*information filtering*).

Um dem Benutzer beim Umgang mit großen Datenmengen zu unterstützen, haben sich zahlreiche Informationsanbieter zur Personalisierung und zur Bereitstellung von persönlichen Empfehlungen hin gewendet. Die wachsende Beliebtheit und den zunehmenden Einsatz kann man an zahlreichenden bekannten Webauftritten sehen (z.B. AMAZON, Yahoo!), am Entstehen von Firmen, die Personalisierungssoftware anbieten (NetPerceptions, Firefly, LikeMinds) sowie an Sonderausgaben von akademischen Zeitschriften (CACM 8/2000), Konferenzen und Workshops (Personalization Summit 2001, ACM SIGIR´99 Workshop on Recommender Systems, IJCAI´99 Workshop on Machine Learning for Information Filtering). Das zeigt, dass Personalisierung sowohl wissenschaftlich als auch praktisch betrachtet als vielversprechender Ansatz im Kampf gegen die Informationsüberflutung gesehen wird.

Im allgemeinen gibt es zwei grundlegende Verfahrensweisen bei der Personalisierung: das inhaltsbasierte und das kollaborative Filtern. Darüber hinaus gibt es hybride Systeme, die beide Techniken miteinander verknüpfen.

3.5.1.1 Inhaltsbasiertes Filtern

In diesem Abschnitt werden die wichtigsten und bekanntesten Projekte bzw. Ansätze vorgestellt, die sich mit inhaltsbasierter Personalisierung bzw. dem inhaltsbasierten Filtern beschäftigen. Die Grundidee besteht darin, die Interessen von Benutzern inhaltlich zu erfassen und zu repräsentieren. Dazu werden die Dokumente ebenfalls inhaltlich betrachtet, um so größere und kleinere inhaltliche Übereinstimmungen zwischen den Dokumenten und den Benutzerinteressen feststellen zu können.

SIFT. Das *S*tanford *I*nformation *F*iltering *T*ool gehört zu den ersten bekannten Systemen mit einer personalisierten Informationslieferung [YM99]. Das System erhält täglich eine riesige Menge (ca. 80000) von Mails aus unterschiedlichen Nachrichtengruppen (USENET) und Mailinglisten und leitet diese nur an einen Benutzer weiter, sofern sie auch für ihn interessant sein könnten. Andernfalls würde der Benutzer von der ungeheuren Flut an Informationen überwältigt werden. SIFT stellt dem Benutzer eine Mail- und eine Webschnittstelle zur Verfügung, über die er seine Profile selbst erstellen und die für ihn als interessant angesehenen Mails (im Folgenden Dokumente genannt) abrufen kann.

Ein Benutzerprofil besteht bei SIFT aus einer Anfrage und einigen Zusatzangaben. Diese Angaben besagen, wie häufig der Benutzer über neue Dokumente informiert werden möchte (meist geben die Benutzer hier täglich an), wie viele Zeilen vom Dokument bei der Benachrichtigung gleich mitangegeben werden sollen und wie lange das Profil gültig sein soll. Anfragen kann der Benutzer entweder boolesch oder gemäß des Vektorraummodells (VRM) formulieren. Bei SIFT sind boolesche Anfragen nur mit AND- und NOT-Verknüpfungen erlaubt. Um die OR-Semantik nachzubilden, kann der Benutzer entsprechend mehrere Profile anlegen. Eine Anfrage, und damit ein Benutzerprofil, bezieht sich also hier auf ein spezielles Interessensgebiet. Für eine Anfrage gemäß des VRM kann der Benutzer eine Menge von Worten und einen Schwellwert zwischen 0 und 1 angeben.

Als Unterstützung für die Formulierung einer VRM-Anfrage steht dem Benutzer eine repräsentative Dokumentenmenge zur Verfügung, auf welcher er die Wirkung der Anfrageterme und des Schwellwerts überprüfen und ggf. noch anpassen kann. Weiterhin kann der Benutzer im laufenden Betrieb jederzeit für ihn interessante Dokumente markieren (*relevance feedback*). Daraufhin modifiziert SIFT die Gewichte der VRM-Anfrageterme im Benutzerprofil.

Bei der Implementierung von SIFT werden ähnliche Verfahren wie bei klassischen IR-Systemen verwendet ([SM83], [Sal89]). Der Unterschied besteht allerdings darin, dass bei SIFT die Profile bzw. Anfragen bekannt sind und die Dokumente ad hoc geliefert werden. Daher indiziert SIFT nicht, wie im IR normalerweise üblich, die Dokumente, sondern die Anfragen. Die Indizierung geschieht durch die Erstellung von Invertierten Listen: Zu jedem Term, der in einer Anfrage auftaucht, wird eine Liste erstellt, welche Verweise auf die

entsprechenden Anfragen enthält. Zusätzlich wird in der Liste auch das Gewicht jedes Terms in der Anfrage notiert.

Für ein neu ankommendes Dokument wird nun die Ähnlichkeit zu den vorhandenen Benutzerprofilen bzw. Anfragen geprüft. Das Dokument und die Anfrage werden jeweils als Vektoren aufgefasst und die Ähnlichkeit über das Kosinusmaß bestimmt. Der normierte Dokumentvektor $D=<w_1, ..., w_m>$ wird mit dem normierten Anfragevektor $Q=<z_1, ..., z_m>$ multipliziert. Also berechnet sich die Ähnlichkeit durch $sim(D, Q) = D * Q = \sum_{i=1}^{m} w_i z_i$. Die Gewichte des Dokumentenvektors w_i und der initialen VRM-Anfrage werden mit TFIDF ebenfalls auf klassische Weise [SM83] berechnet, d.h. die Termhäufigkeit *tf* (*term frequency*) wird mit der inversen Dokumenthäufigkeit *idf* (*inverse document frequency*) multipliziert. Ein Term, der in vielen Dokumenten auftaucht, erhält einen niedrigen idf-Faktor, da dieser Term ein schlechter Diskriminator ist. Da bei SIFT kein fester Dokumentbestand vorliegt, wird zur Bestimmung des idf-Faktors wiederum die repräsentative Dokumentenmenge herangezogen. Für den booleschen Fall werden als Gewichte 1 für die gewünschten und -1 für die nicht gewünschten Terme angegeben.

SIFTER. SIFTER ist ebenso wie SIFT ein System zum personalisierten Filtern von Mails, nämlich von LISTSERV-Mails. Das Akronym SIFTER steht für *S*mart *I*nformation *F*iltering *T*echnology for *E*lectronic *R*esources. Dadurch wird bereits angedeutet, dass hier nicht nur die klassischen IR-Verfahren verwendet werden, sondern auch maschinelle Lernverfahren zum Einsatz kommen.

Das SIFTER-System [MMPL96] unterteilt sich in drei Module: Ein Modul dient zur Modellierung des Dokumentenraums (*document representation module*), ein weiteres Modul bildet Gruppen von Dokumenten (*classification module*) und ein drittes Modul stellt die Verbindung zwischen den Dokumentgruppen und den Benutzerinteressen her (*user profile learning module*). Dieses zweistufige Verfahren reduziert die Komplexität beim Bestimmen der relevanten Dokumente. Allerdings liegt diesem Ansatz die Annahme zugrunde, dass das Benutzerinteresse immer konstant ist für alle Dokumente innerhalb einer Gruppe.

Die Dokumente werden gemäß des klassischen VRM als gewichtete Vektoren dargestellt. Zur Reduzierung der Komplexität werden hier allerdings nicht beliebige Terme zur Repräsentation eines Dokuments zugelassen, sondern ein kontrolliertes Vokabular, nämlich die Einträge eines Thesaurus. Ein Thesaurus enthält in der Regel Schlagworte, die in einem Fachgebiet als sehr relevant und gut diskriminierend angesehen werden können. Hier wird die ACM-Klassifikation für den Bereich der Informatik verwendet (ACM-CSS). Die Gewichte werden mit der bekannten TFIDF-Formel [SM83] berechnet, wobei wie bei SIFT auch eine repräsentative Dokumentenmenge benötigt wird, um die inverse Dokumenthäufigkeit zu bestimmen.

Die Dokumentenvektoren werden nun mit einem unüberwachten Klassifikationsverfahren gruppiert. Als Algorithmus wird die *maximin-distance* Klassifikationstechnik [TG74] verwendet, als Ähnlichkeitsmaß zwischen zwei Vektoren wieder das klassische Kosinusmaß [SM83]. Zu jeder Klasse wird ein repräsentativer Vektor (*centroid*) gebildet. Die entstandenen Klassen sind also nicht identisch mit den vorgegebenen Kategorien der Mails. Ein neues Dokument wird nun mit allen repräsentativen Vertretern der Klassen verglichen und diese Ähnlichkeiten werden dann an das Benutzerprofilmodul weitergegeben.

Als Benutzerprofil wird ein Vektor gehalten, der zu jeder Dokumentklasse einen Relevanzwert enthält. Der Benutzer kann die gelieferten Dokumente jeweils mit 1 (relevant) oder 0 (nicht relevant) bewerten. Der Relevanzwert wird als die Wahrscheinlichkeit berechnet, dass ein Dokument dieser Kategorie interessant für den Benutzer ist. Dazu wird der *pursuit learning algorithm* [MT89] verwendet, ein Reinforcement-Algorithmus, der schnell konvergiert und in der Lage ist, mit einer Gewichtung von Alternativen umzugehen und nicht nur die beste identifizieren kann. Diese Eigenschaft wird für die Sortierung der Ergebnisse benötigt. Konvergenz bedeutet hier, dass der Vektor des Benutzerprofils schrittweise in Richtung eines Einheitsvektors gezogen wird, der nur an der Stelle der interessantesten Klasse eine 1 und an allen anderen Stellen eine 0 aufweist. Darin steckt also die Annahme, dass ein Benutzer primär an einem Thema interessiert ist.

Um auf Veränderungen des Benutzerinteresses reagieren zu können, wurde ein zusätzliches Modul entwickelt [LMMP96]. Es beobachtet für jede Dokumentklasse die Folge der Relevanzwerte und berechnet zu einem gegebenen Relevanzwert mit dem Bayes'schen Verfahren die a-posteriori-Wahrscheinlichkeit, dass ein Interessenswechsel stattgefunden hat. Prinzipiell können bei diesem Ansatz zwei Fälle auftreten: Das bisherige Lieblingsthema wird uninteressant oder ein neues Thema wird zusätzlich interessant. Der zweite Fall ist bei SIFTER schwieriger zu erkennen als der erste. Da die Dokumente gemäß der Relevanz ihrer Kategorie sortiert werden, erhält der Benutzer kaum die Möglichkeit, eine positive Bewertung zu einem in der Liste weiter unten aufgeführten Dokument zu machen. Wurde nun eine Interessensverschiebung beobachtet, wird der Vektor an jeder Stelle mit 1 neu initialisiert, um jeder Kategorie die gleiche Chance zu geben, das neue Hauptinteresse des Benutzers zu werden.

Die Experimente zur Validierung wurden lediglich mit simulierten Benutzern durchgeführt. Dabei entstanden zwölf Klassen von Dokumenten, die mit jeweils 30 Termen beschrieben wurden. Die präsentierten Dokumente verteilten sich gleichmäßig über die entstandenen Klassen. Das Lernen eines Profils benötigte etwa 10 Tage, wenn man einen Benutzer annimmt, der eine sehr präzise Bewertung der Dokumente angibt, d.h. entweder 0,1 oder 0,9 bei allen Dokumenten der entsprechenden Klasse. Bei dem Experiment wurde angenommen, dass sich der Benutzer für vier der zwölf Klassen interessiert. Bei einem nicht so entscheidungsstarken Benutzer wird die Qualität des Profils zwar auch nach 10 bis 20 Tagen deutlich besser, erreicht aber insgesamt eine wesentlich schlechtere Qualität (0,6) als im

ersten Fall (0,8). Der Wert berechnet sich aus dem Anteil der positiven Bewertungen der jeweils ersten drei Dokumente. Ohne das Lernen eines Benutzerprofils erreicht das System einen Wert von weniger als 0,4.

LIBRA. LIBRA ist ein System für die personalisierte Suche nach Büchern. Die Abkürzung steht für *L*earning *I*ntelligent *B*ook *R*ecommending *A*gent [MR00]. Das System kann ausprobiert werden unter http://www.cs.utexas.edu/users/libra/. LIBRA ist für Anfragen geeignet, die große Ergebnismengen zurückliefern. Der Benutzer bewertet die ersten zehn Treffer und daraufhin wird die gesamte Ergebnismenge gemäß der Vorlieben des Benutzers umsortiert.

LIBRA führt eine Datenbasis mit Informationen zu Büchern, die zuvor von amazon.com extrahiert wurden. Dabei wurden bei amazon.com nur Bücher mit verfügbaren Zusatzinformationen berücksichtigt, d.h. Bücher mit mindestens einer Zusammenfassung, einer Rezension oder einem Kundenkommentar. Zu jedem Buch wurden folgende Beschreibungsmerkmale extrahiert: Titel, Autor, Zusammenfassung, Rezensionen, Kundenkommentare, verwandte Autoren, verwandte Titel und Schlagworte. Es wurden zwar auch die Angaben zu ISBN, Datum, Preis, Verlag, usw. extrahiert, diese wurden aber für die Generierung von Empfehlungen nicht verwendet. Für jedes Beschreibungsmerkmal wurde eine ungeordnete Liste von Worten (ohne Stoppworte) gebildet, die Zusammenfassung, Rezensionen und Kundenkommentare wurden unter einem Feld mit der Bezeichnung „Beschreibung“ zusammengefasst. Die aufgebaute Datenbasis enthält Bücher aus vier Bereichen: Erzählungen (3061 Bücher), Science Fiction (3813 Bücher), Krimis (7285 Bücher) und Wissenschaft (6177 Bücher).

Zur Erstellung eines Benutzerprofils muss der Benutzer nun auf seine Anfrage hin zehn Bücher mit Werten von 1 (schlecht) bis 10 (gut) bewerten. Aus diesen Angaben und den Informationen zu den Büchern wird dann sein Benutzerprofil gelernt. Als Personalisierungstechnik verwendet LIBRA die Bayes’sche Textkategorisierung, allerdings wurde sie für diesen Ansatz speziell erweitert vom Umgang mit Wortmengen auf den Umgang mit Vektoren von Wortmengen. Ein Benutzerprofil ist hier eine Liste von denjenigen Beschreibungsmerkmalen, die am ehesten auf eine positive oder negative Bewertung hindeuten. Dazu gibt der Wert an, um wie viel wahrscheinlicher es ist, dass ein bestimmtes Wort in der Beschreibung eines positiv bewerteten Buches auftaucht. Die Zahlenwerte dieser Wahrscheinlichkeiten sind weniger bedeutend, da LIBRA nicht den absoluten Interessantheitswert eines Buches berechnet, sondern eine Reihenfolge erzeugt. Durch zusätzliche Bewertungen im weiteren Betrieb kann das Benutzerprofil jederzeit weiter angepasst werden.

LIBRA erlaubt dem Benutzer sein Profil einzusehen. Zudem kann der Benutzer sich eine Empfehlung erklären lassen. Dann gibt LIBRA die früheren Bewertungen des Benutzers an, die diese Wahrscheinlichkeit am meisten beeinflusst haben.

Im allgemeinen waren die Benutzer nach 20 Bewertungen mit den Top-3 und Top-10 Empfehlungen von LIBRA sehr zufrieden (Bewertungen von über 0,8). In den Experimenten wurde zudem untersucht, inwieweit die kollaborative Ausrichtung der Merkmale „verwandte Titel“ und „verwandte Autoren“ zur Qualität der Empfehlungen beigetragen haben. Dazu wurden die gleichen Experimente auch ohne Berücksichtigung dieser beiden Merkmale durchgeführt. Die Experimente lieferten zwar keine deutlich schlechteren, aber dennoch statistisch signifikant schlechtere Ergebnisse. Das deutet bereits auf das Potential der kollaborativen Ansätze hin und legt auch zugleich die Vermutung nahe, dass eine Kombination der inhaltlichen und kollaborativen Filterverfahren eine weitere Verbesserung gegenüber den Einzelverfahren bringen könnte.

ResearchIndex. ResearchIndex, vormals unter dem Namen CiteSeer als Forschungsprototyp am NEC Research Institute entstanden, ist ein Recherchesystem im Bereich der Informatik, welches Forschungsartikel, die im elektronischen Volltext im WWW verfügbar sind, aufspürt, herunterlädt, indiziert und nutzbar macht [GBL98]. Bei der Indizierung werden die Referenzen eines Artikels in besonderem Maße berücksichtigt, so dass zusätzlich ein Netz zwischen allen Artikeln aufgespannt wird, welches die aktiven *zitiert*- und auch die passiven *wird-zitiert-von*-Beziehungen enthält.

Auch ResearchIndex wurde mit Personalisierungstechniken ausgestatten [BLG99]. Die Personalisierung hat hier zum Ziel, dass der Benutzer zu einem bestimmten Thema auf dem Laufenden bleiben kann. Dazu wird ein Benutzerprofil angelegt, welches aus zwei verschiedenen Teilen besteht. In einem Teil können Bedingungen (*constraint matching*) an neu eintreffende Dokumente angegeben werden, z.B. dass das Dokument bestimmte Schlagworte enthalten oder bestimmte Papiere zitieren soll. Dabei können auch Bedingungen an die URL von neu eintreffenden Dokumenten gestellt werden. Diese Option ist sinnvoll, wenn die aktuellsten Publikationen von bestimmten Forschergruppen beobachtet werden sollen. Diese werden i.d.R. auf einem bestimmten Webserver zur Verfügung gestellt. Im zweiten Teil des Benutzerprofils können interessante Dokumente angegeben werden, von denen die Ähnlichkeit mit neu ankommenden Dokumenten überprüft wird. Die Ähnlichkeit kann sich auf den Volltext, aber auch auf die gemachten Zitierungen beziehen. Der Benutzer hat die volle Kontrolle über sein Profil, d.h. er kann es einsehen, selbst umformulieren und frei gestalten.

Die Ähnlichkeit von zwei Dokumenten wird mit dem TFIDF-Ansatz und dem Kreuzprodukt der beiden Vektoren berechnet. Der Schwellwert ist vom System fest vorgegeben. Von jedem Dokument wird sowohl die Kurzfassung als auch der gesamte Text separat repräsentiert. Dokumente werden als ähnlich betrachtet, wenn sie eine ähnliche Kurzfassung oder einen ähnlichen Textkörper aufweisen. Die andere Form der Ähnlichkeit bezieht sich auf die Literaturreferenzen am Ende der Dokumente. Dokumente werden als ähnlich betrachtet, wenn sie dasselbe Dokument zitieren. Dabei ist die Ähnlichkeit umso größer, je unbekannter das zitierte Papier ist. Wird ein sehr zentrales und bekanntes Dokument zitiert, wird die Aussagekraft wie beim TFIDF-Ansatz geringer eingeschätzt.

OBIWAN. OBIWAN liefert den Projektrahmen, in dem man sich unter anderem auch mit der Personalisierung der Informationssuche beschäftigt [PG99]. Als Suchwerkzeug wird hier die Meta-Suchmaschine ProFusion verwendet. Da Suchmaschinen oft Ergebnismengen mit mehr als 1500 Treffern zurückliefern und selbst von den ersten 20 Treffern nur die Hälfte von den Benutzern als relevant beurteilt werden, sollen durch den Einsatz von Personalisierungstechniken die Ergebnisse gefiltert und die Sortierung der Ergebnisse verbessert werden.

Der vorgeschlagene Ansatz arbeitet mit der Klassifikation von Dokumenten und der Bestimmung der Benutzerinteressen in Bezug auf diese Dokumentklassen. Zur Textklassifikation wird die Magellan-Begriffshierarchie verwendet, welche aus ca. 4400 hierarchisch angeordneten Konzepten bzw. Begriffen besteht. Die Dokumente, die unter jedem Begriff eingeordnet sind, werden dazu verwendet, um jedes Konzept in der klassischen Vektordarstellung zu repräsentieren. Dafür werden alle Dokumente unter einem Begriff in ein Superdokument zusammengemischt. Die darin vorkommenden Worte und die über die klassische TFIDF-Formel bestimmten Gewichte liefern die Vektordarstellung zu jedem Konzept.

Die Benutzerprofile sind ebenso hierarchisch strukturiert und enthalten zu jedem Konzept einen Interessenswert. Die Benutzerprofile werden automatisch, also ohne explizite Mitarbeit des Benutzers, erzeugt. Dazu wird lediglich das Such- und Navigationsverhalten des Benutzers betrachtet. In regelmäßigen Abständen wird die Browserhistorie analysiert. Die gefundenen Dokumente werden mit den vorhandenen Begriffsvektoren auf Ähnlichkeit hin überprüft. Damit werden die fünf ähnlichsten Kategorien zu jedem Dokument identifiziert und der Interessenswert der zugehörigen Konzepte wird erhöht. Dazu wird die berechnete Ähnlichkeit zusätzlich noch mit dem Verhältnis von Betrachtungszeit zu Seitenlänge multipliziert, wobei sich die Länge der Seite eher als vernachlässigbar herausgestellt hat. In ersten Experimenten wurden in den meisten Benutzerprofilen etwa 50-200 Kategorien gefunden, die einen Interessenswert größer als 0 hatten. Unter den 10 bzw. 20 relevantesten Kategorien waren etwa die Hälfte der Kategorien auch tatsächlich relevant für den Benutzer.

Stellt der Benutzer nun eine Anfrage an ProFusion, so werden die Ergebnisse, genau genommen die Titel und die Zusammenfassungen der gefundenen Seiten überprüft: Das durchschnittliche Benutzerinteresse an den Top-5-Kategorien des Dokuments wird bestimmt. Der neue Rang des Dokuments berechnet sich als Produkt aus dem von ProFusion vorgegebenen Rang und dem bestimmten Benutzerinteresse an dem Dokument. In Experimenten konnten damit die Precision- und Recall-Werte um bis zu 8 % verbessert werden. Das Filtern der Ergebnisse erbrachte wenig Verbesserungen, da die Nicht-Relevanz mit dem gewählten Ansatz nur schwer zu bestimmen ist.

PAWS/Yarrow. PAWS (*P*ersonalized *A*daptive *W*eb *S*earch, [CM01]) ist ein Ansatz zur Personalisierung der Suche im Web und ist Teil des Yarrow-Projekts [CM00]. Der Benutzer

kann für seine Suche eine von acht Suchmaschinen (z.B. Altavista, Northern Light, Yahoo!) auswählen.

Bei PAWS wird das Benutzerprofil automatisch erstellt: Auf dem Rechner des Benutzers werden sämtliche Textdokumente betrachtet. Die n häufigsten Worte werden zusammen mit der Häufigkeit ihres Vorkommens als Benutzerprofil (P) gespeichert.

Bei einer Suche werden aus Performanzgründen nur die ersten 40 gefundenen HTML-Seiten (U) abgerufen und auf Ähnlichkeit mit dem Benutzerprofil überprüft. Die Ähnlichkeit S wird hier mit folgender Formel berechnet:

$$S = \sum_{i=1}^{n} \frac{w_i(U)}{w_i(P)}$$

wobei w_i die Anzahl der Vorkommen vom i-ten Wort des Benutzerprofils angibt. Je nach tolerierbarer Wartezeit kann der Benutzer die Anzahl der gefundenen Dokumente beliebig variieren. Dem Benutzer werden nun die zehn besten und die zehn schlechtesten Treffer präsentiert, welche er jeweils positiv oder negativ bewerten kann. Allerdings zeigt das System dem Benutzer lediglich die URL der Dokumente an. Um ein Dokument beurteilen zu können, muss der Benutzer also die Seite in einem separaten Fenster anschauen. Der Benutzer kann beliebig viele der 20 angezeigten Dokumente bewerten. Drückt er anschließend den Feedback-Knopf, werden die Gewichte aller Dokumente mit dem TW2-Algorithmus (Tailored Winnow 2, [CM00]) entsprechend angepasst und damit wird die Präsentationsreihenfolge neu berechnet. Der Benutzer kann ein solches Feedback so oft geben, bis er mit der Reihenfolge zufrieden ist. Dann betätigt er den Knopf zum Zeigen aller Ergebnisse.

Experimente haben ergeben, dass die Benutzer mit dem Ergebnis nach fünf Iterationen bei einer Bewertung von insgesamt 17 Dokumenten sehr zufrieden sind.

AIS. AIS ist ein System für die personalisierte Weiterleitung von aktuellen Nachrichtenmeldungen aus neun verschiedenen Sparten [BP00]. Die Abkürzung steht für *A*daptive *I*nformation *S*erver. Dabei stehen dem Benutzer zwei Möglichkeiten zur Verfügung, um auf das System zuzugreifen: eine Webschnittstelle für normale Browser bzw. Rechner und eine Schnittstelle für PDAs (Persönliche Digitale Assistenten).

Das Benutzerprofil besteht aus zwei Teilen. Ein Teil repräsentiert kurzfristige, der andere Teil langfristige Benutzerinteressen. Kurzfristig können Themen oder Diskussionen verfolgt werden, die den Benutzer momentan interessieren und über die er weiter informiert werden möchte. Allerdings können diese Themen oder Diskussionen dann auch relativ plötzlich abgeschlossen sein.

Das kurzfristige Profil enthält die letzten 100 bewerteten Nachrichten mit ihrem gesamten Text. Soll nun eine neue Nachricht bewertet werden, werden zunächst die Ähnlichkeiten zu den gespeicherten Nachrichten berechnet. Dazu werden die Nachrichten als gewichtete

Vektoren dargestellt, deren Gewichte mit TFIDF und deren Abstand mit dem Kosinusmaß berechnet werden. Die nächsten Nachbarn sind die, deren Abstand kleiner als t_min ist. Ist der Abstand zu einem Dokument sogar kleiner als t_max, wird angenommen, dass die Nachricht sozusagen zu ähnlich und damit bereits bekannt ist und nicht empfohlen wird. Wird zu einer Nachricht kein nächster Nachbar gefunden, kann sie gemäß des kurzfristigen Profils nicht bewertet werden und wird an das langfristige Profil weitergeleitet.

Um das langfristige Benutzerprofil zu erstellen, werden zu jeder Nachrichtensparte die Dokumente indiziert, anschließend werden geeignete Terme (*feature selection*) zur Reduktion der Vektordimensionen identifiziert. Gewählt werden solche Terme, die häufig in Dokumenten vorkommen, aber anhand derer sich die Dokumente auch gut unterscheiden lassen. AIS arbeitet mit 150 aussagekräftigen Termen pro Nachrichtensparte. Mit dem naiven Bayes-Algorithmus kann nun für jeden Benutzer die Wahrscheinlichkeit gelernt werden, mit der ein Dokument, wenn es gewisse aussagekräftige Terme enthält, interessant für den Benutzer ist. Bei diesem Ansatz können also, wie auch schon beim kurzfristigen Benutzerprofil, mehrere Interessensgebiete durch dasselbe Benutzerprofil ausgedrückt werden.

Durch Experimente konnte herausgefunden werden, dass jedes Profil alleine betrachtet schlechtere Precision- und Recall-Werte liefert als die Kombination von kurz- und langfristigem Benutzerprofil. Bei der Webschnittstelle wurde eine explizite Benutzerbewertung gefordert, über die PDA-Schnittstelle wurde die Bewertung implizit aus dem Leseverhalten des Benutzers abgeleitet. Je weiter der Benutzer im Text einer Nachricht liest, desto höher wird der vergebene Interessantheitswert der Nachricht gewählt. Bei den Experimenten lieferten die expliziten Bewertungen bessere Ergebnisse als die impliziten. Im Rahmen dieses Projekts wurde auch eine Funktionalität entwickelt, die die berechneten Relevanzwerte dem Benutzer erklären kann [BP99].

3.5.1.2 Kollaboratives Filtern

Die zweite Form der Personalisierung ist das kollaborative Filtern. Die Idee des kollaborativen Filterns besteht darin, die Interessen und das Verhalten der Allgemeinheit bzw. der Nutzerschaft für Empfehlungen im Einzelfall heranzuziehen. Diese Ansätze sind extrem wirksam und wurden auch sofort in kommerzielle Produkte integriert. Aus vielen Forschungsprototypen sind mittlerweile Firmen geworden. Da die Unterschiede der verwendeten Techniken nicht so groß sind, werden hier nur die zwei bekanntesten Ansätze und einige kommerzielle Produkte sowie deren Einsatzgebiete vorgestellt.

GroupLens. GroupLens war der erste und ist sicherlich auch der bekannteste Ansatz zum kollaborativen Filtern von Mails aus Newsgruppen [KMMH97].

GroupLens führt eine zweigeteilte Datenbasis. In einem Teil werden sämtliche Bewertungen gespeichert (*ratings*), die die Benutzer über bisherige Nachrichten abgegeben haben. Als Bewertungen werden die Zahlen 1 (schlecht) bis 5 (gut) zugelassen. Beim kollaborativen

Filtern treten Benutzerprofile nicht so explizit in Erscheinung wie beim inhaltsbasierten Filtern. Dafür werden Ähnlichkeiten zwischen den Benutzern berechnet, indem ihre bisherigen Bewertungen miteinander verglichen werden. Dazu wird die Korrelation der Bewertungen ermittelt. Die Korrelationswerte (r) liegen zwischen 1 (maximale Übereinstimmung) und -1 (entgegengesetzte Meinung). Unter der Annahme, dass i Nachrichten von beiden Benutzern K und L bewertet worden sind, berechnet sich die Korrelation (auch Pearson-Koeffizient genannt) von Benutzer K und L durch

$$r_{KL} = \frac{\sum_i \left(K_i - \overline{K}\right)\left(L_i - \overline{L}\right)}{\sqrt{\sum_i \left(K_i - \overline{K}\right)^2}\ \sqrt{\sum_i \left(L_i - \overline{L}\right)^2}}$$

wobei K_i die Bewertung der i-ten Nachricht von Benutzer K und $\overline{K}$ der Durchschnittswert über sämtliche Bewertungen des Benutzers K ist [RISB94]. Die Korrelationen aller Benutzerpaare werden alle 24 Stunden neu berechnet. Die Benutzerdaten werden gemäß ihrer Korrelation geclustert abgespeichert.

Soll nun die Relevanz ($K_{j\text{-pred}}$) einer neuen Nachricht j für den Benutzer K ermittelt werden, so werden alle Benutzer M, die diese Nachricht bereits bewertet haben, herangezogen und deren Bewertungen M_j mit der Korrelation zu Benutzer K (r_{KM}) folgendermaßen verrechnet:

$$K_{j\text{-pred}} = \overline{K} + \frac{\sum_{M \in \text{Bewerter}} \left(M_j - \overline{M}\right) r_{KM}}{\sum_{M \in \text{Bewerter}} \left| r_{KM} \right|}$$

Der errechnete Relevanzwert dient zur Sortierung der neuen Nachrichten für den Benutzer, wobei jeweils eine Frage und alle Antworten auf diese Frage als Einheit betrachtet und in der Reihenfolge nicht auseinander gerissen werden. Eine Einheit wird gemäß ihrer relevantesten Nachricht in die Ergebnismenge einsortiert.

Die Technologie von GroupLens wird auch für Filmempfehlungen verwendet (MovieLens, http://www.movielens.umn.edu). Anhand dieser Anwendung beschäftigt man sich damit [HKR00], wie die Transparenz von Empfehlungen für den Benutzer erhöht werden kann. Dazu werden dem Benutzer seine Bewertungen gezeigt, die den größten Einfluss auf die Relevanzberechnung der Empfehlungen haben. Darüber hinaus werden dem Benutzer die Bewertungen der Nachbarn angezeigt, so dass er sich einen Eindruck davon machen kann, auf welcher Datengrundlage der Relevanzwert errechnet wurde. Als einfache Form der Darstellung wird die Aufschlüsselung gewählt, wie viele Nachbarn den Film mit welchem Wert klassifiziert haben. Als etwas weniger intuitiv verständliche, aber aussagekräftigere Form wird die Darstellung als Histogramm verwendet, in dem die Bewertung der Nachbarn in Abhängigkeit zur Nähe der Nachbarn zum Benutzer aufgetragen wird. In Experimenten sahen 86% der Benutzer die Erklärungen zu den Empfehlungen an. Durch den Einsatz dieser Erklärungskomponente konnte die Akzeptanz des Systems deutlich verbessert werden.

Ringo. Ringo ist ein am MIT entwickeltes System zur personalisierten Empfehlung von Musikalben und Künstlern [SM95]. Dazu müssen die Benutzer ihren Musikgeschmack beschreiben. Das geschieht, indem ein Benutzer beim ersten Kontakt mit dem System eine Liste von 125 Künstlern, oder zumindest Teile davon, bewertet. Die Auswahl der 125 Künstler wird teilweise zufällig getroffen, damit Bewertungen von allen vorhandenen Künstlern entstehen können. Teilweise werden auch die Künstler aufgeführt, die von den meisten anderen Benutzern bereits bewertet worden sind, da dadurch schnell die Ähnlichkeit zu anderen Benutzern festgestellt werden kann. Ringo verwendet ebenfalls keine explizite Modellierung von Benutzerprofilen, sondern setzt diese mit den Bewertungen gleich. Zusätzliche Bewertungen von Künstlern, Alben oder einzelnen Musikstücken können jederzeit abgegeben werden, so dass sich der Musikgeschmack mit der Zeit auch verfeinern oder verändern kann. Im allgemeinen geht Ringo allerdings davon aus, dass der Musikgeschmack im Normalfall von langfristiger Natur ist und abrupte Veränderungen äußerst selten vorkommen.

Der Benutzer kann Ringo nach neuen Alben und Künstlern fragen, die er mögen wird, oder nach Alben und Künstlern, die ihm nicht gefallen werden. Ebenfalls kann er zu einem bestimmten Künstler oder Album die Prognose für ihn erfragen.

Um Vorhersagen machen zu können, wird zunächst die Ähnlichkeit zu den anderen Benutzern festgestellt. Anschließend werden deren Bewertungen umgekehrt proportional zu ihrer Ähnlichkeit gewichtet und miteinander verrechnet. Die Bewertungsskala besteht bei Ringo aus 7 Stufen. Studien zuvor hatten ergeben, dass die Genauigkeit bei mehr als 7 Wahlmöglichkeiten nicht höher wird [RR91]. Zur Berechnung der Ähnlichkeit von Benutzern sieht Ringo vier verschiedene Verfahren vor:

a) die Berechnung der mittleren quadratischen Abweichung der Bewertungen,

b) den klassischen Pearson-Koeffizient (r), der im Gegensatz zu a) auch eine negative Korrelation berücksichtigen kann,

c) einen beschränkten Pearson-Koeffizient, der nur größer wird, wenn beide Bewertungen gleichzeitig größer oder kleiner als 4 sind, und damit wiederum auch nur positive Korrelationen (negative Korrelationen traten nur ganz selten auf) betrachtet,

d) die Künstler-Künstler-Ähnlichkeit, die aufgrund der vorhandenen Bewertungen nicht die Ähnlichkeit zwischen den Benutzern, sondern zwischen den Künstlern und Alben berechnet (mit dem beschränkten Pearson-Koeffizient).

Experimente mit Ringo verglichen die vier Algorithmen in Bezug auf Präzision und Menge der ableitbaren Empfehlungen. Im allgemeinen verhielten sie sich recht ähnlich, der beschränkte Pearson-Koeffizient stellte sich als der beste heraus, wohingegen die Qualität der Empfehlungen bei der quadratischen Abweichung und dem Künstler-Künstler-Algorithmus höher war, aber insgesamt weniger Empfehlungen angegeben werden konnten. Mehr als die

Wahl der Algorithmen erwies sich die Menge der vorhandenen Daten als ausschlaggebend, zum einen die Anzahl der verfügbaren Künstler und Alben (2500), zum anderen die Anzahl der Benutzer (und damit der abgegebenen Bewertungen, >200) und schließlich auch die Menge der abgegebenen Bewertungen von einem einzelnen Benutzer (100-200 zu Beginn und später mehr).

Kommerzielle Produkte. Die zwei bekanntesten und verbreitetsten Produkte im Bereich des kollaborativen Filterns bieten die Firmen NetPerceptions Inc. und Firefly Network Inc an. Beide sind aus der Kommerzialisierung der oben erwähnten Forschungsprototypen entstanden. NetPerceptions basiert auf der Technologie von GroupLens und wird beispielsweise bei amazon.com, CDNow.com oder Moviefinder.com eingesetzt. Firefly basiert auf der Technologie von Ringo und ist hinter Barnes and Nobles, The Rolling Stone und Yahoo! zu finden. Firefly wurde 1998 von Microsoft aufgekauft und ist mittlerweile in deren Produktpalette aufgegangen. Es gibt auch noch zahlreiche weitere Firmen und Produkte zum kollaborativen Filtern, beispielweise Autonomy oder LinkMinds.

Die Anwendungen, die mit einer kollaborativen Personalisierungstechnik ausgestattet werden, sind allerdings nicht so vielfältig. Hauptsächlich werden kollaborative Empfehlungen im Unterhaltungsbereich, im speziellen für Bücher, Musik und Filme verwendet. Sofern eine elektronische Kaufmöglichkeit mit dem System verbunden ist, können neben den Bewertungen der Benutzer auch Angaben über das Kaufverhalten für die Empfehlungen genutzt werden, wie beispielsweise bei AMAZON. Die Daten zum Kaufverhalten haben den Vorteil, dass sie sehr aussagekräftig sind und dem Benutzer keinen Mehraufwand abverlangen.

Auf der Modellierungsebene hat das kollaborative Filtern große Vorteile. Die Dokumente, um die es geht, müssen nicht mehr explizit mit ihren Merkmalen beschrieben werden. Dadurch können Bücher, CDs, Spiele und diverse andere Produkte gemeinsam empfohlen werden.

3.5.1.3 Hybride Systeme

Hybride Systeme setzten zur Personalisierung der Informationen sowohl inhaltsbasierte als auch kollaborative Techniken ein, um dadurch die Schwächen der einzelnen Techniken ausgleichen und die Stärken kombinieren zu können.

Bei inhaltsbasierten Systemen werden Dokumente aufgrund von inhaltlichen Übereinstimmungen mit dem Profil vorgeschlagen. Dadurch sind Texte recht gut geeignet, oder eben Objekte, von denen eine inhaltliche Beschreibung existiert. Musikempfehlungen wären beispielsweise auf rein inhaltlicher Basis nur schwer zu generieren. Bei inhaltsbasierten Systemen muss ein Benutzerprofil erst erstellt werden, bevor die Unterstützung für den Benutzer beginnen kann (*new user ramp-up problem*).

Bei kollaborativen Systemen besteht das Problem, dass nur Dokumente empfohlen werden können, wenn für sie Bewertungen existieren (*new item ramp-up problem*). Weiterhin hängt

die Qualität der Empfehlungen von der Anzahl und Güte der nächsten Nachbarn ab, d.h. Individualisten werden von kollaborativen Systemen schlecht unterstützt (*gray sheep problem*). Bei kollaborativen Techniken müssen die Gegenstände der Empfehlungen nicht explizit modelliert werden, sondern nur die Bewertungen über sie geführt werden. Die Stärke dieser Ansätze liegt daher in themen- und auch bereichsübergreifenden Empfehlungen, die mehr oder weniger eine Frage des persönlichen Geschmacks sind. Hier muss der Benutzer zuerst einige Bewertungen abgegeben haben, bevor er individuelle Empfehlungen beziehen kann (*new user ramp-up problem*).

WebWatcher. WebWatcher ist ein System, das den Benutzer beim Navigieren durch das Web begleitet [JFM97]. WebWatcher führt den Benutzer ausgehend von einer zentralen Webseite, in diesem Fall von der Einstiegsseite von CMU's Informatikfakultät, durch die Webseiten der Fakultät, indem das System dem Benutzer jeweils die nächsten Verweise (Links) vorschlägt. Dem Benutzer wird die originale Webseite präsentiert, auf die empfohlenen Verweise wird der Benutzer mit davor und dahinter eingefügten Icons in Form von zwei Augen aufmerksam gemacht. Pro Seite werden maximal drei Verweise empfohlen. Die Vorschläge generiert WebWatcher sowohl aus den Interessensangaben des Benutzers als auch aus dem beobachteten Verhalten früherer Besucher.

Zu Beginn der Navigation muss der Benutzer über Stichworte sein Interesse bzw. sein Ziel angeben. Diese Stichworte bilden sozusagen das Benutzerprofil, welches im Verlauf der Navigation nicht verändert wird. Von nun an werden dem Benutzer Webseiten präsentiert, die durch Empfehlungen von Verweisen angereichert sind. WebWatcher verfolgt zwei verschiedene Ansätze, um die Relevanz von Verweisen zu berechnen: Ein Ansatz reichert jeden Verweis mit den Stichworten der Besucher an, die diesen Verweis in der Vergangenheit verfolgt haben (ANNOTATE). Der zweite Ansatz reichert jeden Verweis mit den Worten an, die auf den nächsten Seiten auftauchen werden, und versucht damit nicht nur die nächste Verweisempfehlung, sondern die Folge der nächsten Verweisempfehlungen zu optimieren. Dazu wird ein Reinforcement-Verfahren verwendet (RL). Die Ähnlichkeit vom Benutzerprofil und den angereicherten Verweisen wird jeweils über das TFIDF-Maß berechnet. Die empfohlenen Verweise werden von WebWatcher sofort geladen und vorgehalten, damit sie dem Benutzer ohne lange Wartezeiten zur Verfügung stehen.

In Experimenten wurden die beiden vorgestellten Ansätze zusammen mit weiteren Vorgehensweisen untersucht und verglichen. Die RANDOM-Strategie schlägt Verweise nach dem Zufallsprinzip vor. POPULARITY empfiehlt die Verweise, die insgesamt am häufigsten verfolgt wurden, also unabhängig vom aktuellen Benutzerinteresse. MATCH berücksichtigt für die TFIDF-Berechnung lediglich die Worte, mit denen der Verweis auf der Webseite aufgeführt wird. Und COMBINE stellt eine Mischung aus ANNOTATE, RL, POPULARITY und MATCH dar, die durch Anwendung einer logistischen Regression ermittelt wurde. Die Kombination berücksichtigt also allgemeine Häufigkeiten, Verhaltensweisen von Benutzern mit ähnlichen Interessen sowie inhaltliche Beschreibungen der Verweise selbst und auch der

nachfolgenden Seiten. Schließlich wurden noch Empfehlungen von Experten zum Vergleich herangezogen (HUMAN). Tabelle 3.7 gibt an, wie viele der tatsächlich verfolgten Verweise auch vom System zuvor empfohlen wurden.

Strategie	Genauigkeit
RAMDOM	31,3 %
POPULARITY	41,9 %
MATCH	40,5 %
ANNOTATE	42,2 %
RL	44,6 %
COMBINE	48,9 %
HUMAN	47,5 %

Tabelle 3.7: Erfolgsquoten verschiedener Strategien für Verweisempfehlungen

Die kombinierte Strategie hat sich dabei als beste Vorgehensweise erwiesen, die zudem noch eine Qualität von Empfehlungen liefert, die mit Expertenempfehlungen gleichzusetzen ist.

Fab. Fab ist ein System zum Empfehlen von Webseiten. Der Ansatz wurde im Rahmen des Stanford Digital Libraries Projekts entwickelt [BS97]. Fab verwendet sowohl inhaltliche als auch kollaborative Personalisierungstechniken. Der Benutzer kann die zehn besten Webseiten zu seinem Profil anfordern.

Ein Benutzerprofil besteht bei Fab aus einem gewichteten Termvektor. Zur Verwaltung der Benutzerprofile sieht die Fab-Architektur *selection agents* vor, welche die Benutzerprofile speichern und aufgrund von Relevanzbewertungen anpassen. Für die Anpassung der Benutzerprofile wird Rocchios Algorithmus [Roc71] verwendet. Die Relevanzkategorien werden in Fab natürlichsprachlich repräsentiert und intern dann auf ganzzahlige Werte von 3 bis -3 abgebildet. In der Nacht werden die Gewichte der Benutzerprofile jeweils mit 0,97 multipliziert, um den Verfall bzw. die Veränderung der Benutzerinteressen zu unterstützen.

Die Dokumente, also die Webseiten, werden ebenfalls durch einen gewichteten Termvektor dargestellt, der die Dimension 100 hat. Zur Repräsentation der Dokumente wird zunächst eine Wortstammreduktion durchgeführt und anschließend werden die Stoppworte entfernt. Die Gewichte werden nach der TFIDF-Formel berechnet. Für jedes Dokument werden dann die 100 Terme mit den größten Gewichten berücksichtigt. Experimente hatten zuvor gezeigt, dass die optimale Performanz mit 30 bis 100 Termen erzielt werden kann und mehr als 100 Terme ein System mit überwachten Lernmethoden übertrainieren [Bal97].

Das Suchen und Indizieren von Webseiten wird von sogenannten *collection agents* durchgeführt. Diese suchen permanent die besten Seiten zu den gegebenen Profilen und machen diese dem System an zentraler Stelle (*central repository*) bekannt. Die Agenten

aktualisieren ihr Suchprofil kontinuierlich gemäß der Bewertungen der Benutzer. Die Suchprofile der Agenten decken in ihrer Gesamtheit die Interessensprofile der Benutzer ab, jeder Agent spezialisiert sich allerdings zunehmend auf einen Themenbereich. Agenten, deren Dokumente von den Benutzern eher schlecht bewertet werden, werden kontinuierlich durch Agenten ersetzt, die besser bewertete Dokumente liefern. Jeder Suchagent spezialisiert sich also auf ein Thema, ein Benutzerprofil kann durchaus mehrere Themen umfassen.

Das zentrale Behältnis enthält zu jedem Zeitpunkt die besten Webseiten der Suchagenten und wird von den Auswahlagenten durchsucht, wenn ein Benutzer seine persönlichen Empfehlungen anfordert. Die Ähnlichkeit zwischen Benutzerprofil und Dokumenten wird dann über das Kosinusmaß berechnet. Gleichzeitig fließen auch die am besten bewerteten Dokumente der nächsten Nachbarn in die Empfehlungen mit ein. Dem Benutzer werden zu seiner Anfrage jeweils zehn Webseiten empfohlen, die er zum einen noch nicht gesehen hat und die zum anderen keine zwei Seiten vom selben Webserver beinhalten.

Knowledge Pump. Im Rahmen des Knowledge Pump Projekts entstand der hier vorgestellte Ansatz zum Umsortieren von Suchergebnissen [CGG00]. Dabei werden nicht nur die Interessen des Benutzers selbst berücksichtigt, sondern auch die Interessen der Gemeinschaften, denen sich der Benutzer zugehörig fühlt.

Der Ansatz betrachtet sowohl Benutzer- als auch Gruppenprofile. Beide Arten von Profilen werden als gewichtete Termvektoren repräsentiert. Abgeleitet werden sie aus einer Dokumentbasis. Dies kann ein Dokumentenmanagementsystem für eine Arbeitsgruppe sein, vielleicht sogar ein in einer Gruppe verwendetes System für Dokumentempfehlungen oder es können die Dokumente eines verteilten Dateisystems bzw. persönliche Dokumente wie Bookmarks oder Mails sein. Um ein Gruppenprofil zu berechnen, werden die gewichteten Benutzerprofile der Gruppenmitglieder addiert und normiert. Die Gewichte können beispielsweise nach dem Expertenzustand des Mitglieds vergeben werden, d.h. nach der Häufigkeit, mit der seine Empfehlungen von den anderen Gruppenmitgliedern befolgt werden.

Vor einer Suche kann der Benutzer seine Identität oder eine für die Suche gewünschte Gruppenzugehörigkeit bekannt geben. Die Suche kann dann mit einer Suchmaschine oder auch mit einer Meta-Suchmaschine durchgeführt werden. Die Ergebnisse werden danach anhand des gegebenen Profils bewertet und umsortiert. Dazu kann entweder die URL und die Kurzfassung einer HTML-Seite herangezogen werden oder aber der gesamte Text, was sich allerdings negativ in einer langen Antwortzeit bemerkbar macht. Durch Bewertungen der Ergebnisse können die Benutzerprofile angepasst werden. Dazu wird der Rocchio-Algorithmus verwendet. Die Veränderung der Benutzerprofile zieht automatisch eine entsprechende Veränderung der Gruppenprofile nach sich.

Der wirkungsvollste Einsatz der Gruppenprofile ergibt sich, wenn eine Arbeitsgruppe zusätzlich ein System zur Empfehlung von Dokumenten betreibt, in dem Dokumente und die bisherigen Bewertungen enthalten sind und welches kontinuierlich um neue, gut bewertete

Dokumente ergänzt werden kann. So muss die Relevanz eines Dokuments nicht immer inhaltlich ermittelt werden, sondern kann aus vorherigen Bewertungen abgeleitet werden.

Recommender. Recommender [BHC98] ist ein hybrider Ansatz zum Empfehlen von Filmen.

Der Benutzer kann Filme auf einer Skala von 1 bis 10 bewerten. Daraus wird ein individueller Schwellwert berechnet, so dass ¼ der Werte darüber und ¾ der Werte darunter liegen. Über mengenwertige Attribute wird nun für jeden Benutzer festgehalten, welche Filme er mag und welche Filme er nicht mag. Ebenso wird zu jedem Film die Menge der Benutzer gespeichert, die diesen Film mögen und die diesen Film nicht mögen.

Auf diesen mengenwertigen Attributen wird *Ripper* angewendet, ein induktives Lernverfahren, welches Regeln lernen kann, anhand derer entschieden werden kann, ob ein Benutzer einen bestimmten Film mögen wird oder nicht. Recommender setzt seinen Schwerpunkt nicht auf Ähnlichkeitsberechnungen, sondern auf Klassifikationen. Dazu werden Entscheidungsbäume und Regeln, vergleichbar mit dem ID3-Verfahren, erzeugt [Coh96].

Zusätzlich werden inhaltliche bzw. beschreibende Eigenschaften zu den Filmen aus der Internet Movie Database entnommen. Dort sind zu jedem Film etwa 16 Attribute aufgeführt, z.B. Schauspieler, Regisseur, Produzent, Autor, Schlüsselworte, Soundsystem, Laufzeit oder Genre. Recommender verwendet lediglich die Genre-Angabe und betrachtet davon auch nur die drei häufigsten Ausprägungen (Komödie, Drama und Aktion). Zu jedem Genre wird wieder ein mengenwertiges Attribut gespeichert, welches die Benutzer enthält, die in ihren Vorlieben größtenteils Filme von dieser Kategorie haben.

Zur Evaluierung wurden einige Experimente durchgeführt. Die Bewertung erfolgt wie im IR üblich durch die Ermittlung der Precision- und Recall-Werte, d.h. der Anzahl der richtigen Empfehlungen und dem Finden aller Filme, die der Benutzer mögen könnte. Hier wurde eine möglichst hohe Präzision angestrebt, da meist nur ein oder wenige Filme empfohlen werden sollen. In den Experimenten wurden verschiedene Varianten von Recommender miteinander verglichen, alle unter Verwendung von Ripper. Mit rein kollaborativen Techniken (p: 77%, r: 27%) wurde ein besseres Ergebnis erzielt, als wenn die verfügbaren inhaltlichen und beschreibenden Eigenschaften der Filme (p: 73%, r: 33%) mitberücksichtigt wurden. Mit der alleinigen Hinzunahme des Genres wurden die besten Ergebnisse erzielt (p: 83%, r: 34%).

3.5.1.4 Resümee

Zunächst sollen die vorgestellten Ansätze anhand der zu Beginn des Kapitels aufgestellten Anforderungen bewertet werden.

Bewertung

A1 (Einheitliche, integrierte Benutzungsschnittstelle): Eine einheitliche und integrierte Benutzungsschnittstelle ist wohl bei jedem der hier vorgestellten Ansätze vorhanden, da jeweils auch ein einheitliches Recherche- bzw. Nachrichtensystem zugrunde liegt. In dieses wurden

die Bewertungen und Empfehlungen jeweils auch nahtlos eingegliedert. Die Ergebnispräsentation bei PAWS zeigt lediglich die URL von einem Treffer an. Um eine HTML-Seite beurteilen zu können, muss der Benutzer in jedem Fall einen zusätzlichen Verweis verfolgen, auch wenn die Wartezeit durch das Vorhalten der Seiten im Cache gering sein wird.

A2 (Einfache, intuitive Benutzerinteraktion): Die meisten der beschriebenen Ansätze arbeiten mit Benutzerbewertungen. Diese sind für kollaborative Ansätze von zentraler Bedeutung, bei inhaltsbasierten Ansätzen dienen sie meist zur Anpassung der Benutzerprofile, da Bewertungen für die meisten Benutzer leichter zu formulieren sind als inhaltliche Angaben zu ihren Interessen. Die vorgestellten Ansätze arbeiten mit unterschiedlichen Bewertungsskalen, diese reichen von 2-stufigen (SIFTER, PAWS) über 5-stufige (GroupLens), 7-stufige (Ringo, Fab) bis hin zu 10-stufigen Skalen (LIBRA, Recommender).

Um dem Benutzer zusätzliche Eingaben zu ersparen, werden die Benutzerprofile bei OBIWAN, PAWS (zu Beginn) und der PDA-Version von AIS automatisch erstellt, d.h. ohne Benutzerinteraktion. Zur Generierung eines Profils werden Informationen über Dokumente und Lesezeiten des Benutzers herangezogen. Ebenso ohne Benutzerinteraktion kommen Systeme aus, die gleichzeitig die Bestellungen oder Käufe der Benutzer abwickeln.

A3 (Ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit): Nur wenige der vorgestellten Ansätze erlauben dem Benutzer, sein Benutzerprofil einzusehen oder bei kollaborativen Ansätzen seine bisherigen Bewertungen anzuschauen und ggf. zu verändern. Eigentlich erlaubt nur ResearchIndex das Einsehen und eine direkte Manipulation des Profils. Die Zielgruppe von ResearchIndex besteht allerdings auch aus Informatikern, von denen ein gewisses Geschick in der Konfiguration erwartet werden kann. LIBRA, AIS und GroupLens liefern auf Wunsch zumindest Erklärungen, aufgrund von welchen Angaben die Empfehlungen generiert wurden. SIFT bietet eine Unterstützung beim Formulieren von Benutzerprofilen an, genauer gesagt eine repräsentative Testdatenmenge, auf der der Benutzer die Wirkung seines Benutzerprofils testen kann.

A4 (Berücksichtigung menschlicher Eigenschaften): Im allgemeinen unterstützen die vorgestellten Personalisierungsansätze sowohl die Individualität des Benutzers als auch eine gewisse Bandbreite an Gestaltungsmöglichkeiten. Etwas speziellere Vorlieben, die nicht so sehr im Trend liegen, können von kollaborativen Ansätzen allerdings schlecht unterstützt werden. Die vorgestellten Personalisierungsansätze helfen dem Benutzer beim langfristigeren Verfolgen seiner Interessen und stellen in Bezug auf die Benutzerinteraktion keine besonderen Anforderungen an die Fähig- und Fertigkeiten des Benutzers. Die Ungeduld des Benutzers wird zu Beginn der Erstellung eines Benutzerprofils notgedrungen strapaziert, dafür können im weiteren Verlauf relevantere Ergebnisse gefiltert bzw. weiter oben in der Ergebnisliste aufgeführt werden, was wiederum im Sinne des Benutzers ist.

Die Kriterien bei der Berücksichtigung menschlicher Eigenschaften können bei Personalisierungsansätzen noch erweitert werden, da die vorgestellten Ansätze jeweils die

Interessen bzw. den Geschmack des Benutzers berücksichtigen. Es ist eine große Herausforderung, die Interessen oder den Geschmack des Benutzers exakt zu fassen und zu modellieren. Inhaltsbasierte Ansätze beziehen die Interessen des Benutzers auf bestimmte Themen. Die Ähnlichkeit von Benutzerinteresse und Dokument wird mehr oder weniger aufgrund von gleichen Wortvorkommen bestimmt. Ein Dokument wird als umso relevanter für einen Benutzer eingeschätzt, je größer die textuelle Ähnlichkeit zu seinem Benutzerprofil ist. AIS unterscheidet zwischen kurzfristigen und langfristigen Benutzerinteressen. Die meisten Benutzerprofile verändern sich kontinuierlich während des Betriebs über Benutzerbewertungen (OBIWAN, LIBRA). SIFTER hat spezielle Vorkehrungen für die leichte Erkennung und Unterstützung von Interessensverschiebungen getroffen. Die Profile von SIFT und ResearchIndex sind in Form von Anfragen abgelegt, diese ändern sich in der Regel nicht. LIBRA, PAWS, OBIWAS können in einem Benutzerprofil mehrere Interessen beschreiben, wohingegen bei SIFT ein Benutzer eine Reihe von Profilen benötigt.

Den kollaborativen Ansätzen liegt die Annahme zugrunde, dass Personen mit ähnlichen Interessen auch über unbekannte Dinge die gleichen Meinungen haben. Man kann dadurch Empfehlungen auch themen- und anwendungsübergreifend generieren. Ohne zusätzlich verfügbare Informationen zu den zu empfehlenden Gegenständen ist dieses Vorgehen zweifelsfrei ein Gewinn. Allerdings zeigen die Anwendungsbereiche der vorgestellten Ansätze auch, dass ihre Stärken eher im Unterhaltungsbereich liegen.

A5 (Vollständige Anfrageplanung und -bearbeitung): Die Antworten sind beim Einsatz von Personalisierungstechniken meist eben nicht vollständig, da ja unrelevante Dokumente herausgefiltert und nur die interessanten Nachrichten weitergeleitet werden sollen oder lediglich einige interessante Dokumente empfohlen werden sollen. LIBRA, OBIWAN, PAWS und Knowledge Pump sind Ansätze, die die Suche unterstützen. Sie verändern die Reihenfolge der Ergebnisse so, dass sie besser den Interessen des Benutzers entspricht. LIBRA und OBIWAN arbeiten vollständig, der PAWS-Ansatz arbeitet auf den ersten n (standardmäßig n=40) Dokumenten vollständig, d.h. der Benutzer bekommt alle vom Suchdienst gefundenen Ergebnisse auch präsentiert, zwar an einer anderen Stelle, aber kein Ergebnis geht verloren. WebWatcher arbeitet ebenfalls vollständig, alle regulären Verweise einer Webseite werden dem Benutzer angeboten, nur die empfohlenen werden deutlicher hervorgehoben.

Kollaborative Ansätze arbeiten in dieser Hinsicht bewusst nicht vollständig, da sie nur einige gute Empfehlungen geben wollen (vgl. Recommender). Sie arbeiten aber auch nicht vollständig, was die Empfehlung von Außenseitern angeht. Diese werden im Verhältnis zu populären Objekten fast nie vorgeschlagen, da wenig Bewertungen für sie existieren.

A6 (Duplikatbehandlung): Von den inhaltsbasierten Ansätzen enthält lediglich der AIS-Ansatz Vorkehrungen, damit dasselbe Dokument nicht mehrfach empfohlen wird. Zudem

wird nicht nur dasselbe, sondern werden auch sehr ähnliche Dokumente dem Benutzer nicht mehr präsentiert.

Kollaborative Ansätze empfehlen in der Regel keine Dokumente, die bereits vom Benutzer bewertet worden sind. Fab merkt sich beispielsweise die gemachten Empfehlungen, so dass keine Empfehlung doppelt vorgeschlagen wird. Zudem bereinigt Fab die zehn empfohlenen Webseiten in der Hinsicht, dass keine zwei Seiten vom selben Server vorgeschlagen werden.

A7 (Schnelle Antwortzeiten): Empfehlungen müssen beispielsweise nicht immer sofort, sondern können auch einmal täglich verschickt werden (SIFT). Um die Ergebnisse schneller berechnen zu können, beschränken inhaltsbasierte Ansätze teilweise die Dimensionen der Termvektoren. Dadurch kann die Komplexität der Ähnlichkeitsberechnungen reduziert, zum Teil sogar die Qualität erhöht werden (SIFTER, AIS). Zur Reduzierung der Komplexität wird teilweise auch mit Gruppenbildung gearbeitet, so dass die Interessen des Benutzers für eine Gruppe von Dokumenten errechnet wird und von einem Dokument nur noch die Gruppenzugehörigkeit bestimmt werden muss, um daraus die Relevanz für den Benutzer ableiten zu können (SIFTER, OBIWAN). Bei kollaborativen Ansätzen können die Ähnlichkeitsberechnungen zwischen den Benutzern auch separat oder in periodischen Abständen durchgeführt werden. Meist werden dann ähnliche Benutzer oder Objekte geclustert abgelegt, so dass aktuelle Berechnungen recht schnell erfolgen können (GroupLens).

Ansätze, die die Suche des Benutzers unterstützen und das Ergebnis umsortieren (LIBRA, OBIWAN, PAWS, Knowledge Pump), betrifft diese Anforderung am stärksten. Soll der gesamte Text der gefundenen HTML-Seiten zur Berechnung der Relevanz verwendet werden, nimmt die Antwortzeit für den Benutzer deutlich zu. Daher verwenden die meisten Ansätze nur die sofort verfügbare Kurzfassung der Webseite. PAWS argumentiert, dass die damit erzielten Resultate nahezu genauso gut sind. Knowledge Pump bemerkt, dass die entstehenden Termvektoren zur Repräsentation doch sehr unterschiedlich sind. Auch wenn Uneinigkeit über die Bewertung der Alternativen herrscht, ist man sich einig, dass man den Forderungen der Benutzer nach schnellen Antwortzeiten in jedem Fall nachkommen muss.

A8 (Berücksichtigung von Lastvorgaben): Die vorgestellten Personalisierungsansätze berichten von keinen Lastvorgaben seitens der zugrundeliegenden Systeme.

Resümee

Betrachten wir nun die vorgestellten Personalisierungstechniken im Hinblick auf eine Anwendung im Literaturbereich. Der Fokus der vorliegenden Arbeit liegt auf der Unterstützung der wissenschaftlichen Literaturrecherche, d.h. es soll in erster Linie die Suche des Benutzers unterstützt werden. Dabei soll gleichzeitig verhindert werden, dass der Benutzer von einer Informationsflut überwältigt wird.

Die Ansätze PAWS und Knowledge Pump, die beide die Informationssuche personalisieren, haben nochmals unsere Anforderung bekräftigt, dass diese Form der Unterstützung als äußerst

zeitkritisch einzustufen ist. Der zusätzliche Aufwand entsteht durch die Nachbearbeitung der Ergebnisse und verlängert somit die Antwortzeit. Die Antwortzeit ist im Hinblick auf Benutzerakzeptanz allerdings genauso wichtig wie die Verbesserung durch Personalisierungstechniken. Leider haben die vorgestellten Ansätze weder konkrete Angaben zur Antwortzeit noch zur Benutzerakzeptanz gemacht.

Die Verwendung einer inhaltsbasierten Personalisierungstechnik ist bei der Literaturrecherche recht naheliegend. Unser Ansatz soll allerdings auf einem Meta-Recherchesystemen aufsetzen, das Bibliotheken, Verlage und Buchhändler zusammenführt. Diese enthalten in der Regel keine Volltexte der Dokumente, sondern im besten Fall einige Beschreibungstexte, wie sie auch von LIBRA verwendet werden. Benutzerstudien haben ergeben, dass neben den inhaltlichen Eigenschaften von Büchern für Benutzer auch andere Angaben relevant sind, beispielsweise Preis, Verfügbarkeit, Sprache, u.ä. Diese Angaben werden in den vorgestellten Ansätzen bisher nicht berücksichtigt.

Kollaborative Personalisierungsansätze sind zweifelsfrei sehr wirksam, fördern allerdings eher Massentrends und sind für Bereiche geeignet, in denen es mehr um den persönlichen Geschmack als um nachvollziehbare Entscheidungskriterien geht. Im Hinblick auf den wissenschaftlichen Bereich der Literatur ist allerdings Individualität und die Vielfältigkeit von Themenbereichen erwünscht. Es gibt beispielsweise Studien von Bibliotheken, die besagen, dass ein Großteil der Bücher von nur wenigen Personen genutzt wird (Pareto-Prinzip, 80:20-Regel). Dadurch würden die Bewertungen bei kollaborativen Ansätzen hauptsächlich von der Meinung einer Minderheit geprägt werden. Bei der Entwicklung des vorgeschlagenen Ansatzes wird daher vom Einsatz kollaborativer Techniken abgesehen.

In den vorangegangenen Abschnitten sind auch die Arten und Eigenschaften der Benutzerinteressen deutlich geworden, die von den verschiedenen Techniken und Ansätzen unterstützt werden können. So kann ein Benutzer durchaus mehrere Interessen gleichzeitig haben, vielleicht sind einige davon kurzfristiger, andere langfristiger Natur. Die meisten vorgestellten Ansätze unterstützen die kontinuierliche Veränderung der Benutzerprofile, manche sogar in besonderem Maße.

Die automatische Bestimmung, von welcher Qualität ein erkanntes Benutzerinteresse nun ist, ist jedoch sehr schwierig. Das automatische Entfernen von Duplikaten bzw. sehr ähnlichen Dokumenten bei den Empfehlungen kann beispielsweise manchmal gewünscht sein (wenn sich der Benutzer beispielsweise für einen Reiseführer zu einem bevorstehenden Urlaub entschieden hat, benötigt er keinen zweiten mehr), bei einem eher Fan-mäßigen Interesse allerdings (beispielsweise in Bezug auf Harry Potter) möchte der Benutzer sicherlich über jedes neu erschienene Buch informiert werden. Zur differenzierteren Beschreibung der Profile sollte der Benutzer dem System zumindest einige Hinweise geben können. Wenn beispielsweise sein Interesse erloschen ist, möchte der Benutzer bestimmt nicht weitere zehn negative Bewertungen zu diesem Thema abgeben müssen.

Nur in wenigen der vorgestellten Ansätze kann der Benutzer das über ihn erzeugte Profil einsehen oder gar beeinflussen. Seltene oder unerfahrene Benutzer würde das sicherlich auch überfordern, für die regelmäßige Arbeit mit dem System und den Interessensprofilen wäre allerdings zumindest die Möglichkeit dazu wünschenswert. Zudem würde es eine gewisse Nachvollziehbarkeit der ausgelösten Aktionen unterstützen, die beispielsweise bei Benutzerbefragungen zu GroupLens bereits eingefordert worden sind.

Wenn man von Personalisierung spricht, darf das Thema Datenschutz und Wahrung der Privatsphäre nicht fehlen. Im Bereich der traditionellen Bibliotheken findet man bemerkenswert wenig Ansätze zur Personalisierung. Die Erklärung dafür wird auf Nachfragen bei Bibliotheken schnell geliefert: Bibliotheken sind nämlich seit je her ein Verfechter des Datenschutzes gewesen. So halten sie beispielsweise auch nur die aktuellen Ausleihdaten ihrer Benutzer und entfernen diese, sobald die Bücher wieder zurückgegeben wurden. Auf jeden Fall ist für die Konzeption eines eigenen Personalisierungsansatzes zu berücksichtigen, dass Benutzer in der Regel keine persönlichen Daten über sich preisgeben wollen. Die vorgestellten Ansätze arbeiten meist mit Pseudonymen, über die sich die Benutzer dem System zu Erkennen geben. Sonst erfordern bzw. verwenden die vorgestellten Ansätze keine Angaben (wie beispielsweise Alter, Geschlecht, Beruf) über den Benutzer. Die einzige Ausnahme bilden hier die Systeme, die eine Kauf-Funktion integriert haben und dazu die vollständigen Benutzerdaten benötigen.

3.5.2 Graphische Ergebnisvisualisierungen

Es ist nicht anzunehmen, dass es „die“ ideale Benutzungsschnittstelle für alle Benutzer einer Digitalen Bibliothek geben wird. Aber es kann durchaus für verschiedene Benutzergruppen unterschiedliche Zugangsformen in Koexistenz geben, zwischen denen der Benutzer wählen oder die er sogar selbst anpassen kann.

Die Fragestellung lautet für diesen Abschnitt daher nicht, welche Klassen von Benutzungsschnittstellen besser sind, Desktop-Metaphern, abstrakte räumliche oder konkrete räumliche Metaphern. Es ist vielmehr interessant zu sehen, welche Arten von Benutzungsschnittstellen es gibt, welche Ideen und Absichten dahinter stehen, welche Vorteile und Nachteile sie mit sich bringen und durch welche Besonderheiten sie sich auszeichnen. Abschließend soll beurteilt werden, welche Art von Benutzungsschnittstelle für das angestrebte Meta-Recherchesystem bzw. die formulierte Problemstellung sinnvoll ist.

Die Forschung beschäftigt sich nun schon seit einiger Zeit mit der Suche nach „geeigneten Benutzungsschnittstellen für das IR“. Hier sollen nun die wesentlichen Arten von Ansätzen vorgestellt werden. Bei den Ansätzen, die eine graphische Präsentation der Ergebnisse einsetzen, geht es hauptsächlich um die Darstellung des Dokumenten- bzw. des Informationsraums. Dabei gibt es abstrakte und konkrete Metaphern, realisiert in 2D oder 3D. Sie ermöglichen eine höhere Darstellungsdichte als die reine Textform, d.h. durch graphische

Mittel können mehr Attribute oder Eigenschaften zu einem Dokument dargestellt werden. Weiterhin sind für den Benutzer bei der Betrachtung verschiedene Perspektiven möglich (wechselnder Fokus bzw. Kontext), sowie Variationen im Level-of-Detail (LOD). Die Ansätze gehen davon aus, dass es eine menschliche Stärke ist, sich in Räumen schnell zu orientieren und ggf. Dinge an derselben Stelle wiederzufinden. Diese Stärke soll für die vorliegende Such- und Orientierungsaufgabe des Benutzers ausgenutzt werden.

Eine andere Gruppe von Ansätzen bleibt bei der Desktop-Metapher bzw. benutzt weiterhin eine gewohnte WIMP-Benutzungsschnittstelle (*W*indow, *I*con, *M*enu, *P*ointing device) und stellt die Ergebnisse hauptsächlich in textueller Form dar, allerdings ergänzt und aufbereitet mit einigen graphischen Darstellungsmitteln.

Bei jedem der im Folgenden vorgestellten Ansätze ist zusätzlich zu berücksichtigen, ob und wie die Praxistauglichkeit und der Erfolg der vorgeschlagenen Technik nachgewiesen wurde.

3.5.2.1 Abstrakte räumliche Darstellungen

Die Mehrheit der existierenden graphischen Benutzungsschnittstellen verwendet eine abstrakte Metapher. Dabei werden die relevanten Dimensionen des Suchraums räumlich dargestellt, um so den Zugang zu den Dokumenten zu erleichtern. Die Ähnlichkeit von Dokumenten zeigt sich in ihrer Nähe zueinander.

LyberWorld. LyberWorld ([Hem95], [HKW94]) ist eine graphische Benutzungsschnittstelle für das probabilistische Volltext Retrieval System INQUERY [CCH92]. Die Benutzungsschnittstelle verwendet die Metapher der räumlichen Navigation in abstrakten Informationsräumen. Hier wird der Informationsraum von den Volltexten der Dokumente gebildet. LyberWorld unterstützt die Suche von Dokumenten, die Navigation im Dokumentenraum und auch das Anschauen von Dokumenten. Dafür werden dem Benutzer drei Werkzeuge zur Verfügung gestellt.

Die Dokumente sind jeweils mit Termen verschlagwortet. Intern liegt eine netzartige Struktur vor, in der jedes Dokument mit seinen Schlagworten und jedes Schlagwort mit den Dokumenten, für die es vergeben wurde, verbunden ist. Dieses Netz wird nun in Form eines Kegelbaums (*NavigationCone*) visualisiert. Die Wurzel des Baums stellt den Begriff dar, nach dem der Benutzer gesucht hat. Nun bilden immer abwechselnd Dokumente (rote Farbgebung[5]) und Schlagworte (blaue Farbgebung[5]) eine tiefere Ebene des Baumes, durch den der Benutzer beliebig navigieren kann (s. Abbildung 3.3). Die helleren Farbtöne zeigen dem Benutzer jeweils an, dass er diesen Bereich des Baumes bereits besucht hat.

[5] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

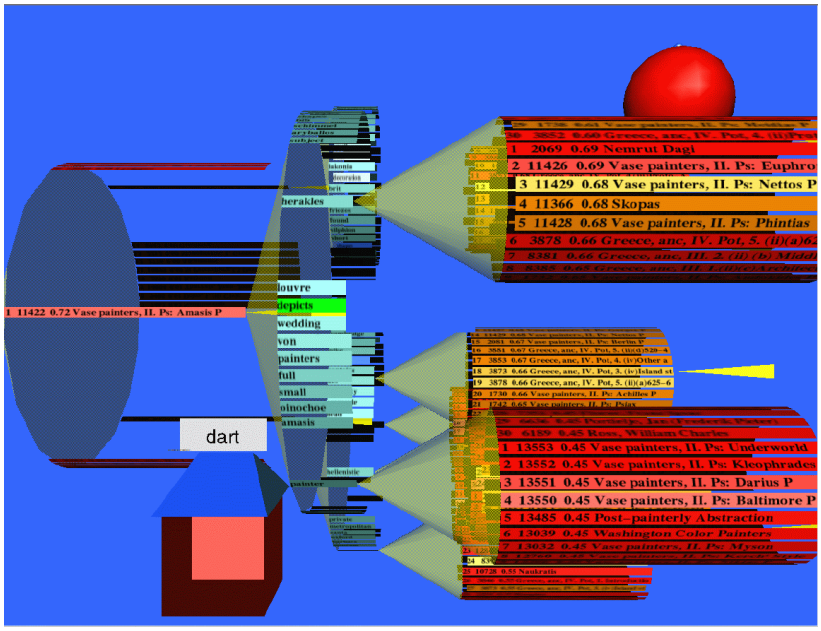


Abbildung 3.3: Darstellung eines Kegelbaums in LyberWorld

Als zweites Hilfsmittel steht dem Benutzer eine Relevanzsphäre (*RelevanceSphere*) zur Verfügung. Das ist eine Kugel, auf deren Oberfläche die Schlagworte, die der Benutzer im Laufe seiner Recherche angesehen und verfolgt hat, als Referenzpunkte gleichmäßig verteilt sind. Innerhalb der Sphäre sind die bisher betrachteten Dokumente in Form von Büchersymbolen jeweils mit dem entsprechenden Abstand zu den Begriffen angeordnet. Diese Referenzpunkte kann der Benutzer nun mit seiner Maus auf der Kugeloberfläche verschieben und in der Stärke ihrer Anziehungskraft verändern, bis er dadurch eine Teilmenge von Dokumenten identifizieren kann, die relevant für ihn ist (s. Abbildung 3.4).

Schließlich stellt LyberWorld noch ein Betrachtungswerkzeug für die Dokumente selbst zur Verfügung (*InformationRoom*), mit welchem der Benutzer den Volltext der Dokumente ansehen kann. Die zur Zeit nicht benutzten Werkzeuge werden dem Benutzer jeweils als kleine Symbole angezeigt. Realisiert wurde LyberWorld in C++ zusammen mit dem IRIS Inventor Toolkit der Firma Silicon Graphics auf einer Grafikworkstation der gleichen Firma.

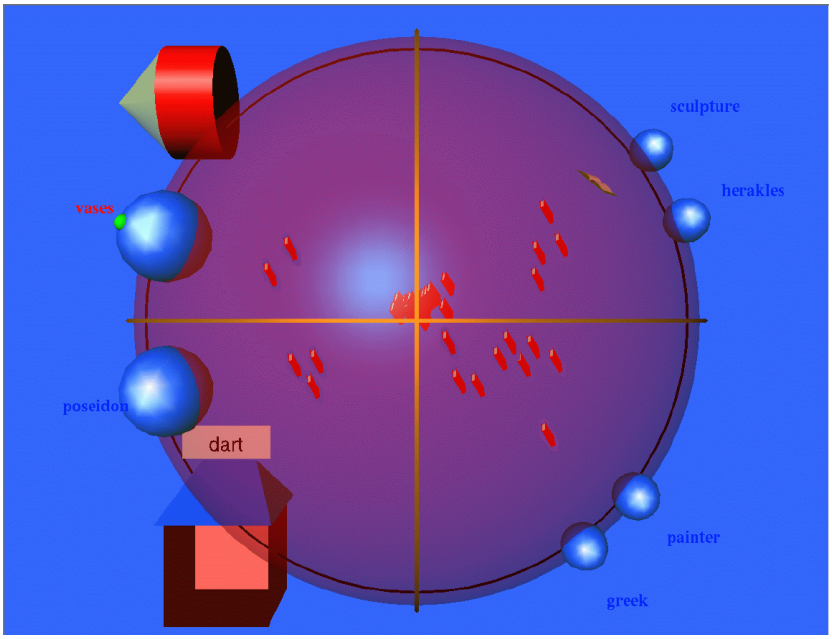


Abbildung 3.4: Darstellung einer Relevanzsphäre in LyberWorld

Die Grundidee von LyberWorld besteht darin, die natürliche Fähigkeit des Menschen zur visuellen Wahrnehmung und zur Orientierung im Raum auszunutzen. Es wird argumentiert, dass sich der Benutzer durch diese Form der Visualisierung schneller in der Menge der Ergebnisdokumente orientieren und zurechtfinden kann und somit die kognitive Beanspruchung des Benutzers im Verlauf einer Informationssuche reduziert werden kann. Benutzerstudien, die dies belegen können, wurden allerdings nicht veröffentlicht.

DocumentFinder. Der VRML-DocumentFinder [DP98] ist ein Werkzeug zur dreidimensionalen Darstellung von Ergebnismengen. Es kann sowohl zur Visualisierung der Ergebnisse einer Nachrichtenrecherche verwendet werden (Newscan-Online) als auch zur Darstellung der Ergebnisse einer Literaturrecherche (INFODATA). In beiden Fällen wird die dreidimensionale Visualisierung als zusätzliche Option zu einer normalen textuellen Ergebnisanzeige angeboten.

INFODATA ist eine Literaturdatenbank für das Gebiet der Informationswissenschaft und umfasst ca. 100 000 Dokumentennachweise. Bei Eingabe eines unspezifischen Suchbegriffs können die Ergebnismengen recht groß werden. Als Hilfsmittel generiert INFODATA selbst für Ergebnismengen mit mehr als 30 Dokumenten eine Liste der zehn relevantesten Begriffe aus dem INFODATA-Thesaurus. Durch das Anwählen eines Begriffs wird dem Benutzer eine stark reduzierte Ergebnismenge präsentiert, die aus der UND-Verknüpfung des Begriffs mit dem ursprünglichen Anfrageterm ermittelt wurde.

Liefert INFODATA eine Trefferanzahl zwischen 30 und 200 Dokumenten, wird dem Benutzer zusätzlich die Präsentation mit dem DocumentFinder angeboten. Dieser gruppiert

die Dokumente mit dem Complete-Link-Cluster-Verfahren [Rij79], d.h. die so erzeugten Themengebiete unterscheiden sich von den Begriffen des INFODATA-Thesaurus. DocumentFinder präsentiert dem Benutzer einen mit VRML realisierten dreidimensionalen Informationsraum (s. Abbildung 3.5), durch den er frei navigieren kann. Der Benutzer benötigt dazu in seinem Browser lediglich ein installiertes VRML-Plugin.

Abbildung 3.5: 3D-Darstellung der Rechercheergebnisse mit dem DocumentFinder

Die blauen[6] Podeste sind in einer hierarchischen Baumstruktur angeordnet und stellen jeweils ein erzeugtes Themengebiet dar. Die Dokumente werden als Würfel dargestellt. Ein roter[6] Würfel bedeutet, dass das entsprechende Dokument durch keinen spezifischeren Begriff als das Themengebiet charakterisiert werden kann. Ein gelber[6] Würfel besagt, dass das entsprechende Dokument thematisch noch genauer spezifiziert werden kann und in einem tiefer liegenden Themengebiet noch einmal auftaucht. Der Benutzer kann diesen Informationsraum entweder durch freies Navigieren oder durch systematisches Anfahren vordefinierter Standpunkte erforschen. Die bibliographischen Informationen bzw. der Inhalt eines Dokuments werden bzw. wird in einem separaten Fenster angezeigt.

Die Grundidee von DocumentFinder ist es, die Strukturen und Beziehungen innerhalb von großen Dokumentenmengen deutlich zu machen. Durch die Realisierung in VRML kann das Visualisierungswerkzeug bzw. können die aufbereiteten Darstellungen im Web zur Verfügung gestellt werden. Beide Anwendungen von DocumentFinder waren zumindest zeitweise für die allgemeine Nutzung im Web verfügbar. Ergebnisse zu Benutzerstudien oder zur Benutzerakzeptanz wurden bisher nicht veröffentlicht.

---

[6] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

3.5.2.2 Konkrete räumliche Darstellungen

Ansätze mit konkreten Metaphern setzen auf den Wiedererkennungswert meist von Dingen des alltäglichen Umgangs und die mitgebrachte Erfahrung des Benutzers in diesem Bereich. Dadurch soll sich der Benutzer intuitiv verhalten und mit der Umgebung interagieren können. Wenn man sich an konkreten Metaphern orientiert, können maximal drei Dimensionen zur Darstellung des Informationsraums verwendet werden.

Cave-ETD. Vorgänger dieses Ansatzes ist eine VRML-basierte Benutzungsschnittstelle [Kip97] für die ETD-Sammlung (*E*lectronic *T*hesis and *D*issertation Collection) der Virginia Tech Universität. Diese Sammlung beinhaltet 1507 Volltexte von Diplomarbeiten und Dissertationen dieser Universität. Zur Visualisierung wird die Metapher einer Bibliothek verwendet, in welcher Dokumente in Form von Büchern in Regalen dargestellt werden. Der Cave-ETD Ansatz [NF00] wählt dieselbe Art der Visualisierung, die Interaktion findet allerdings nicht an einem Computerbildschirm statt. Der Benutzer muss sich für eine Literaturrecherche in einen speziellen würfelförmigen Raum begeben, auf dessen Boden und Wände Bilder projiziert werden, so dass der Benutzer den Eindruck vermittelt bekommt, selbst zwischen den Bücherregalen zu stehen und mit ihnen interagieren zu können. Der Benutzer trägt dazu eine spezielle Brille, mit der er die erzeugten dreidimensionalen Effekte wahrnehmen kann. Abbildung 3.6 zeigt die Cave-ETD-Umgebung. Die weißen Linien deuten an, welche Teile der Umgebung auf welche Wand des würfelförmigen Raums projiziert werden.

Für Benutzerexperimente wurde ein Szenario angelegt, das 383 Bücher aus den Fachrichtungen Betriebswirtschaft und Personalwesen enthält. Die Bücher wurden zuvor geclustert, so dass jedes Buch zu genau einem Cluster gehört. Es gibt zwei Regalreihen, wobei jeweils nur die obersten zwei Regalböden mit Büchern bestückt sind (s. Abbildung 3.6). In einem Regalfach stehen etwa 16 Bücher. Die Bücher stehen nach Clustern aufgereiht nebeneinander, aufeinander folgende Cluster werden durch ihre Farbgebung[7] kenntlich gemacht, d.h. die aufeinander folgenden Cluster werden abwechselnd blau und grün dargestellt. Die Bücher haben einen fixen Standpunkt im Regal, d.h. sie ordnen sich nicht aufgrund einer Benutzeranfrage um. Hier wird also eher die Metapher einer Präsenzbibliothek verfolgt. Der Benutzer verfügt über eine 3D-Maus, mit der er sozusagen frei durch den Raum und zu den Regalen laufen und gegebenenfalls Bücher genauer anschauen kann. Der Benutzer kann entweder den Titel und Autor eines Buches oder die ersten 200 Worte der Zusammenfassung anschauen.

[7] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

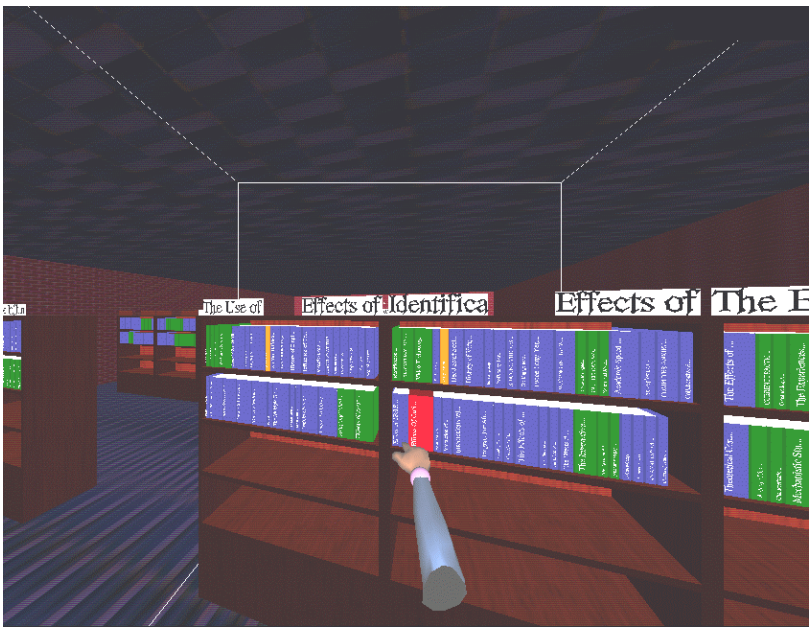


Abbildung 3.6: Benutzerinteraktion in Cave-ETD

Testpersonen für die Experimente waren zehn freiwillige Studenten aus dem Hauptstudium. Die Aufgabe der Probanden war es, das beste Buch zu einem vorgegebenen Thema auswählen. Dabei waren von jedem Teilnehmer sechs Themen zu bearbeiten.

Die Experimente stellen eine Faktoranalyse dar, in der zwei Clusteralgorithmen und drei Visualisierungsarten erprobt werden sollten. In Bezug auf die Clusterung wurden zwei bekannte Verfahren getestet, wobei mit dem Scorpion-Algorithmus viele kleine Cluster, mit durchschnittlich 2,6 Dokumenten, entstanden, mit dem SPIRE-Verfahren wenige große Cluster, mit durchschnittlich 11 Dokumenten. Weiterhin wurden drei Arten der Ergebnisvisualisierung getestet. Die Best-Match-Variante zeigte die gebildeten Cluster und nur das am besten passende Buch in roter Farbe[7]. Die Farbverlauf-Variante zeigte sowohl die Cluster, als auch überlagert eine rot-gelb-Färbung der Dokumente, welche den Grad der Relevanz zur gestellten Anfrage darstellt. Die dritte Variante zeigte nur die relevanten Bücher mit rot-gelb-Färbung, d.h. die umgebenden Bücher wurden ausgeblendet.

Ergebnisse der Studie waren folgende: Die verschiedenen Clustervarianten wurden nicht unterschiedlich genutzt. Die im Cluster benachbarten Bücher wurden zwar betrachtet, allerdings war die Relevanzmarkierung für die Auswahl eines Buches deutlich dominanter als diese Ähnlichkeitseigenschaft. Unabhängig von der Art der Ergebnisvisualisierung wurden etwa gleich viele Bücher inspiziert, allerdings war die Qualität des letztendlich ausgewählten Buches unterschiedlich. Die Best-Match-Variante führte zu schlechteren Ergebnissen, der Unterschied der anderen beiden Varianten war nicht signifikant. Das unterstreicht nochmals den geringen Einfluss der Clusterbildung.

In allen Fällen hörten die Benutzer nach einer gewissen Anzahl von inspizierten Büchern mit der Suche auf und entschieden sich für das beste bis dahin gefundene Buch. Dieses Ergebnis kann auch durch andere Studien bestätigt werden. Benutzer suchen nicht bis zu einem Optimum, sondern entscheiden sich für das relevanteste Dokument, das sie in dem vorgesehenen Zeitrahmen gefunden haben. Allerdings verlangte die gegebene Aufgabe hier auch keine erschöpfende Suche.

Bei den Experimenten wurde entdeckt, dass die Positionierung der Bücher im Regal eine Rolle spielt. Die Bücher auf dem unteren Regalboden wurden beispielsweise nicht immer berücksichtigt. Schließlich wurde noch über Probleme der Benutzer beim Betrachten der Bücher und den Bewegungen im Raum berichtet.

QuakeBib. QuakeBib [CS02] verwendet zur Unterstützung der Literaturrecherche die Metapher der traditionellen Bibliothek. Als Modellierungswerkzeug wurde QuakeII verwendet, ein bekanntes und weit verbreitetes 3D-Computerspiel der Firma id Software. Unter QuakeII kann man mit Hilfe von sogenannten Leveleditoren eigene Welten, d.h. Umgebungen, Räume, Interaktionsformen und Funktionalitäten erstellen.

QuakeBib hat das Gebäude der Universitätsbibliothek Karlsruhe naturgetreu nachgebildet, sowohl von außen als auch von innen. In dieser Umgebung kann sich der Benutzer nun frei umschauen und bewegen. Er findet seine vertraute Umgebung vor: Regale mit Büchern, Türen, Wendeltreppen, Aufzüge, Schließfächer, ein schwarzes Brett und vieles mehr.

Zum Ausführen einer konkreten Suchanfrage kann sich der Benutzer in einen der Rechercheräume begeben. Dort füllt er an einem Rechner wie gewohnt ein Suchformular aus. Als Recherchequellen stehen dem Benutzer die Kataloge der Universitätsbibliothek Karlsruhe (UBKA), der Badischen Landesbibliothek (BLB) und von AMAZON zur Verfügung. Zur Integration dieser Dienste setzt QuakeBib auf dem UniCats-System auf. Die Suchergebnisse werden anschließend plastisch in Form von Büchern in Regalen dargestellt. Die Farben[8] symbolisieren in Abbildung 3.7 die Ausleihbarkeit eines Buches, auf dem Buchrücken erscheint die Sprache, durch die verschiedenen Regale werden die Bücher gemäß ihrer Bezugsquelle gruppiert.

[8] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

Abbildung 3.7: Gebäude der UBKA und Rechercheraum in QuakeBib

Zum besseren Umgang mit großen Ergebnismengen kann der Benutzer nun interaktiv die Visualisierungsform ändern. Über ein Steuerpult an der Wand kann er bestimmte Merkmale (Verfügbarkeit, Bezugsquelle, Sprache) durch bestimmte graphische Darstellungsmittel (Farbe, Symbol auf Buchrücken, Position im Regal) anzeigen lassen.

Mit seinem Laserpointer kann der Benutzer die bibliographischen Angaben jedes Buches abrufen, diese werden am unteren Rand des Bildschirmes eingeblendet. Durch Aktivieren der grünen Pfeile kann der Benutzer die Regalböden bewegen und dadurch sämtliche Bücher eines Regals betrachten.

Die Motivation von QuakeBib lieferte die weite Verbreitung von 3D-Computerspielen, die als Zeichen bzw. als Beweis für die Akzeptanz durch die Benutzer gewertet werden kann. Die Software der Spiele ist sowohl bekannt für intuitive und komfortable Benutzerinteraktionen als auch für eine leistungsfähige 3D-Grafikmaschine. Benutzerstudien zum praktischen Einsatz von QuakeBib sind bisher nicht durchgeführt worden.

MiBiblio. MiBiblio ([FSG00], [SPPC01]) ist ein Ansatz zur Visualisierung von persönlichen Arbeitsbereichen in Digitalen Bibliotheken. MiBiblio basiert auf der Raum-Metapher, d.h. einzelne Benutzer oder auch Benutzergruppen können persönliche Bereiche gestalten mit digitalen Dokumenten oder Dokumentennachweisen, die sie zur Zeit gerade verfolgen, häufiger benötigen, besonders interessieren oder anderen Personen zur Verfügung stellen möchten. MiBiblio liegt der Bestand von digitalen Diplomarbeiten und der Online-Katalog der Physikbibliothek der Universität de las Americas-Puebla zugrunde. Realisiert wurde die Benutzungsschnittstelle als Java Applet.

Der Benutzer kann seinen persönlichen Raum mit Regalen und anderen Einrichtungsgegenständen ausstatten. Die initiale Raumkonfiguration entspricht der Rolle des Benutzers. MiBiblio bietet dafür folgende Rollen an: Student im Vordiplom oder im Hauptdiplom, wissenschaftlicher Mitarbeiter, Professor, Bibliothekar, Angestellter oder Systemadministrator. Abbildung 3.8 zeigt ein Beispiel eines persönlichen Raums. Bücher

repräsentieren beliebige Materialien, Dokumentennachweise, Notizen oder auch Volltexte. Sie können durch Drag&Drop aus anderen Anwendungen in den Raum hineingebracht und dort beliebig in die Regale einsortiert werden. Das animierte Buch auf dem linken Regal steht für den Empfehlungsdienst SyReX, der inhaltsbasierte und kollaborative Empfehlungen generieren kann. Die Eule symbolisiert einen mobilen Suchagenten (Viajerus), der auch digitale Sammlungen von Partnerinstitutionen durchsuchen kann.

Abbildung 3.8: Benutzungsschnittstelle von MiBiblio

Für den Aufbau von Gruppenbereichen ist der Leiter der Gruppe zuständig. So kann ein Professor beispielsweise sämtliche Materialien, die zu seiner Veranstaltung benötigt werden oder hilfreich sein könnten, in Form von Dokumenten, Buchempfehlungen oder auch speziellen Diensten seinen Studenten zur Verfügung stellen.

Es wurden bereits einige Experimente mit MiBiblio zur Performanz und zur Benutzerakzeptanz durchgeführt. Die beobachteten Benutzer waren mit minimaler Anleitung in der Lage, den Raum mit Büchern und teilweise sogar mit Diensten auszustatten. Im allgemeinen fanden 80% der Benutzer die persönlichen und die Gruppenbereiche nützlich. 70% der Benutzer konnten die ihnen gegebenen Aufgaben durchführen und erfüllen.

3.5.2.3 Desktop-Metaphern und WIMP-Benutzungsschnittstellen

In diesem Abschnitt stellen wir Ansätze vor, die auch mit graphischen Mitteln und Darstellungen arbeiten, allerdings in der gewohnten Desktop-Metapher des Benutzers bleiben. Auch diese Ansätze haben neuartige und hilfreiche Darstellungstechniken hervorgebracht.

SenseMaker. SenseMaker [BW97] entstand im Rahmen des Stanford Digital Library Project und setzt auf der Schnittstelle des Stanford InfoBus auf, einer CORBA-basierten Integrationsplattform für heterogene Informationsdienste. SenseMaker bietet eine Rechercheumgebung an, die dem Benutzer eine interaktive Suche bei einer Reihe von

Informationsdiensten ermöglicht, beispielsweise bei der Fachdatenbank INSPEC, der Library of Congress (LoC), NCSTRL und bei diversen WWW-Suchmaschinen.

SenseMaker unterstützt in erster Linie recht breit angelegte Benutzeranfragen, die sich erst im Laufe der Recherche weiter konkretisieren. Ein Beispiel hierfür wäre die Recherche eines Studenten, der Literatur für die Anfertigung einer Seminararbeit zum Thema graphische Benutzungsschnittstellen zusammenstellen möchte. Bei breit angelegten Benutzeranfragen werden die Ergebnismengen in der Regel recht groß und unüberschaubar. Um den Benutzer bei der Begutachtung der Ergebnismenge zu unterstützen, bietet SenseMaker Operationen auf der Ergebnismenge an, die diese in Gruppen (*bundles*) ähnlicher Dokumente zusammenfassen und strukturieren. Die Gruppierung erfolgt standardmäßig anhand des Webservers, der die Dokumente zur Verfügung stellt. Der Benutzer kann aber auch andere Attribute zur Strukturierung von Literaturreferenzen vorgeben, beispielsweise die Jahreszahl oder den Autor.

Für Gruppen von Dokumenten bietet SenseMaker eine Expansionsoperation an. Sie führt automatisch in anderen Diensten eine Suche nach Dokumenten durch, die den Bedingungen der Gruppe entsprechen, und fügt die gefundenen Dokumente der Gruppe hinzu. Weiterhin können Gruppen gelöscht, weiter spezialisiert oder gespeichert werden.

SenseMaker bietet flexible Strategien für die Duplikaterkennung an. Dabei werden vom System im jeweiligen Kontext (bibliographische Beschreibungen, Web-Ressourcen, ...) sinnvolle Duplikaterkennungsstrategien vorgegeben, aus denen der Benutzer eine auswählen kann. Angeboten werden beispielsweise die Titelgleichheit oder die Gleichheit des gesamten Textdokuments.

Die Benutzungsschnittstelle ist nach anfänglichen Versuchen mit einer HTML-basierten Dialogführung nun als Java-Applet realisiert. Für die Visualisierung der Ergebnismenge wurde eine Baumstruktur gewählt, in der Knoten für Dokumentgruppen () und Duplikatgruppen () verwendet werden. Zusätzlich wurde eine neue Darstellungsform entwickelt, die als Mischform zwischen einer tabellarischen Darstellung (für jedes Attribut eine eigene Spalte) und einer rein textuellen Aneinanderreihung von Attributwerten gelten kann, die sogenannten *Hi-Cites* (*highlightable citations,* s. Abbildung 3.9). Hierbei werden die einzelnen Attributwerte einer bibliographischen Beschreibung wie in einem traditionellen Zettelkatalog textuell aneinandergereiht. Verweilt der Benutzer mit dem Mauszeiger jedoch über einem Attributwert, so werden alle Attributwerte desselben Attributs in den anderen bibliographischen Beschreibungen farblich[9] hervorgehoben, so dass der Benutzer recht schnell alle vorkommenden Ausprägungen eines Attributs vergleichen kann.

[9] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

Bates, M.J. The design of browsing and berrypicking techniques for the online search interface. *Online Review* 13,5 (October 1989), pp. 407-24.

D-Lib Magazine. http://www.dlib.org/.

O`Day, V., and Jeffries R. Orienteering in an information landscape: how information seekers get from here to there, in *Proceedings of INTERCHI'93* (Amsterdam, April 1993), ACM Press, pp. 438-45.

Abbildung 3.9: Hi-Cites-Darstellung von bibliographischen Informationen

Im Rahmen von SenseMaker wurden zwei Benutzerstudien mit fünf und drei Informatikstudenten durchgeführt. In der ersten Studie konnte gezeigt werden, dass Benutzer von SenseMaker im Vergleich zu Benutzern desselben Recherchesystems ohne Strukturierungsoperationen weniger oft (in 60 % der Zeit im Vergleich zu 80% der Zeit) eine völlig neue Suche starteten, sondern die angebotenen Operatoren zur Expansion und Spezialisierung nutzten. In der zweiten Studie sollten sich die Benutzer ohne vorherige Einweisung selbständig innerhalb von 45 Minuten mit der Funktionalität von SenseMaker vertraut machen. Dabei wurden Verbesserungsmöglichkeiten für die Gestaltung der Benutzungsschnittstelle identifiziert und in der folgenden Version von SenseMaker entsprechend ergänzt.

DLITE. DLITE (*D*igital *L*ibrary *I*ntegrated *T*ask *E*nvironment, [CPWB97]) ist eine Benutzungsschnittstelle für die Interaktion mit verschiedenartigen Diensten im Bereich der Digitalen Bibliotheken. DLITE verfolgt die Metapher der zentralen Arbeitsstelle (*workcenter*), an der alle Werkzeuge, die man im Rahmen seiner Tätigkeit benötigt, griffbereit sind. Die Werkzeuge werden jeweils aufgabenspezifisch von Experten zusammengestellt und anschließend der Allgemeinheit zugänglich gemacht. Beispielsweise gibt es zentrale Arbeitsstellen für Veröffentlichungs- bzw. Verlegertätigkeiten oder zur Unterstützung der Literaturrecherche und der Informationssuche.

Die Werkzeuge, die dem Benutzer für seine Tätigkeit zur Verfügung stehen, werden Komponenten (*components*) genannt. DLITE unterstützt fünf Arten von Komponenten: Dokumente (), Kollektionen (s. Abbildung 3.10), Anfragen (), Dienste () und Personen (). Dokumente können einfache Dokumentennachweise sein, wie sie üblicherweise von Suchdiensten zurückgeliefert werden, oder auch Dokumente, deren Inhalt – sogar in mehreren Formaten – verfügbar ist. Kollektionen sind Sammelbehälter für andere Komponenten, die häufigste Form der Kollektion ist die Ergebnismenge eines Suchdienstes, die mehrere Dokumente enthält. Führt der Benutzer den Mauszeiger über ein einzelnes Dokument der Kollektion, wird der Titel eingeblendet. Führt der Benutzer einen Doppelklick innerhalb des Kreises aus, so öffnet sich ein Browser und zeigt die Dokumente der Ergebnismenge an. Die Zahlen in der Mitte des Kreises geben an, wie viele Dokumente bei der Suche insgesamt

gefunden wurden und wie viele Dokumente pro Portion abgerufen werden können. Klickt der Benutzer genau auf diese Zahl, so werden die nächsten zehn Dokumente angezeigt. Jede Kollektion erhält automatisch einen Namen, Folio bezeichnet die INSPEC Datenbasis der Universität Stanford. Gleichzeitig wird zu jeder Ergebnismenge auch die Anfrage vermerkt, durch die sie erzeugt wurde.

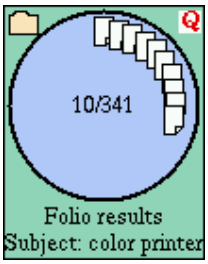


Abbildung 3.10: Kollektion in DLITE

Die Anfragen werden zusätzlich in einer Menge aufbewahrt, so dass die Anfragen wiederverwendet oder mit anderen Benutzern ausgetauscht werden können. Neben Suchdiensten für Literatur oder das Web bietet DLITE auch eine Reihe von anderen Diensten an, beispielsweise die automatische Erstellung von Zusammenfassungen, Formatumwandlungen von Dokumenten oder die Bearbeitung von Bibliographiedaten (InterBib). Um dem Benutzer die Aktivität der Dienste zu signalisieren, werden die Symbole in diesem Zeitraum animiert. Die angebotenen Dienste werden über die einheitliche Schnittstelle, die der Stanford Infobus für jeden Dienst zur Verfügung stellt, in DLITE eingebunden. Schließlich repräsentiert DLITE auch Benutzer, z.B. zur Zugangskontrolle oder Abrechnung von kostenpflichtigen Diensten.

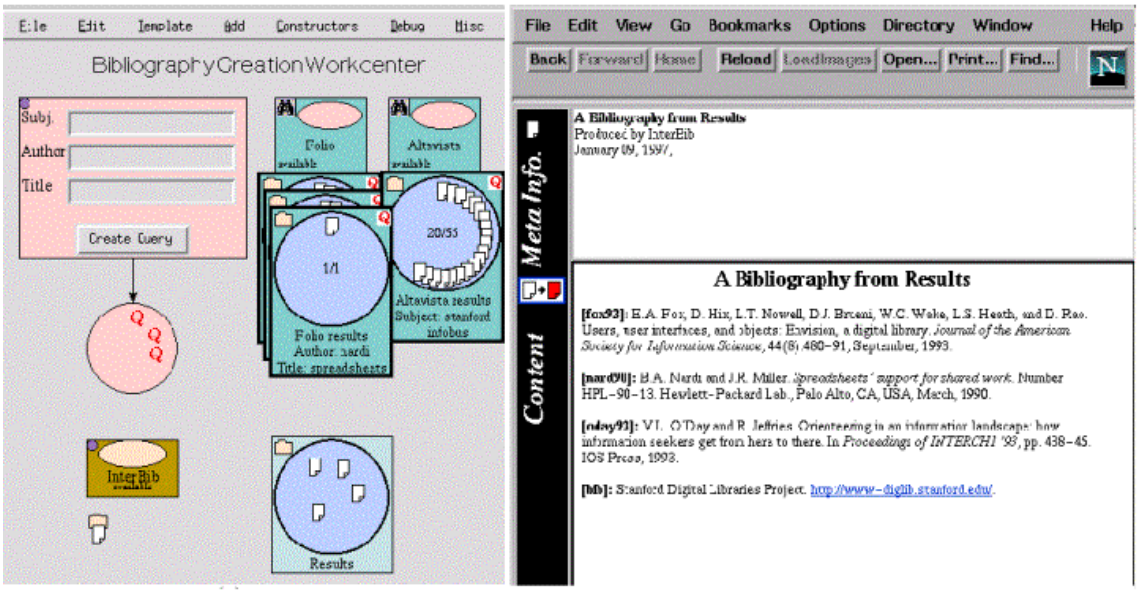


Abbildung 3.11: DLITE-Beispiel zur Erstellung einer Bibliographie

Abbildung 3.11 zeigt eine Arbeitsstelle zur Erstellung einer Bibliographie. Zunächst kann der Benutzer durch mehrere Anfragen eine Reihe von Dokumenten finden. Der Benutzer kann mit dem Drag&Drop-Mechanismus einzelne Dokumente in eine separate Menge (Results) kopieren. Sobald er die gewünschte Menge von Dokumenten zusammengestellt hat, kann er diese Menge als Eingabe an den InterBib-Dienst übergeben, der daraus eine einheitlich formatierte Liste von Literaturreferenzen erstellt. Diese kann durch einen Browser dargestellt werden.

Zur Evaluierung von DLITE wurde eine Benutzerstudie durchgeführt. In dieser Studie sollte herausgefunden werden, ob einige allgemeine Erklärungen für die Benutzung von DLITE ausreichend sind, damit Benutzer spezielle Aufgaben durchführen können. Die sechs Testpersonen, denen das System vorher nicht bekannt war, konnten in weniger als 30 Minuten die vier geforderten Literaturreferenzen finden und daraus eine einheitliche Bibliographie erstellen. Damit ist die Benutzungsschnittstelle in der Tat recht intuitiv zu bedienen, auch wenn bei den Beobachtungen aufgefallen ist, dass der InterBib-Dienst häufig mit einzelnen Dokumenten anstelle von Dokumentenmengen angesprochen wurde. Zwei Benutzer entdeckten sogar den Vorteil, dass bei DLITE auch mehrere Suchen parallel ausgeführt werden können.

Gestaltungsrahmen. In [SBC98] wird ein Gestaltungsrahmen für Benutzungsschnittstellen zur Informationssuche in Textdokumenten präsentiert. Die Arbeit benennt das Problem, dass viele aktuelle Suchschnittstellen unnötig komplex sind und zudem wesentliche Eigenschaften des dahinterliegenden Recherchesystems verschleiern. Viele Systeme verwenden oft überraschende Anfragetransformationen, unvorhersehbare Wortstammreduktionen und rätselhafte Gewichtungen für Anfragefelder. Die Reihenfolge der Ergebnisse ergibt sich aus einem Relevanzwert, dessen Bedeutung für viele Benutzer unbekannt und manchmal auch ein gut gehütetes Geheimnis des Anbieters ist. Dadurch entstehen Verwirrung und Frustration – sowohl bei fortgeschrittenen Benutzern als auch bei Anfängern.

Der Ansatz beschreibt die ideale Rechercheschnittstelle als verständlich, nachvollziehbar, vorhersehbar und kontrollierbar und schlägt einen Gestaltungsrahmen bestehend aus vier Phasen vor, der die Klarheit und Kontrolle des Benutzers erhöhen und die Inkonsistenzen zwischen verschiedenen Benutzungsschnittstellen verringern soll. Die identifizierten Phasen einer Recherche sind Anfrageformulierung (*formulation*), Suchaktion (*action*), Ergebnispräsentation (*review of results*) und Anfrageverfeinerung (*refinement*).

Die Anfrageformulierung ist die komplexeste Phase. Hier sind nämliche diverse Entscheidungen zu treffen: In welchen Quellen soll gesucht werden (*sources*), welche Felder von Dokumenten sollen durchsucht werden (*fields*), nach welchem Text soll gesucht werden (*what to search for*) und welche Varianten dieses Textes sollen akzeptiert werden (*variants*)? Diese Bestandteile einer Anfrage sollen klar aus der Benutzungsschnittstelle hervorgehen.

Besondere Klärung benötigt der Punkt, nach welchem Text gesucht werden soll. Die Angabe des Textes kann Verknüpfungsoperatoren (AND, OR, NOT), Phrasen oder Platzhalter erforderlich machen. Hier muss klar sein, ob Stoppworte wie AND beispielsweise eliminiert werden und auf welche Weise ein Wildcard spezifiziert werden kann (beispielsweise * bei Altavista und $ bei Lycos). Vorgeschlagen wird hier eine Umsetzung, bei der der Suchtext und die Verknüpfungen separat eingegeben werden, alle Eingaben eines Textfeldes werden als Phrase interpretiert. Sobald ein Textfeld ein Stopwort enthält, wird der Benutzer darauf hingewiesen. Zur Klärung der zugelassenen Varianten sollten verschiedene Verfahren explizit vom Benutzer ausgewählt werden können, beispielsweise Beachtung von Groß-/ Kleinschreibung, Wortstammreduktion, Bildung phonetischer Varianten oder Nutzung eines Thesaurus.

Die Suchaktion kann implizit und explizit ausgelöst werden. Anwendungen mit dynamischen Anfragen benötigen keinen Suchknopf, da das Ergebnis kontinuierlich angezeigt und bei Änderung der Suchbegriffe auch sofort aktualisiert wird. Bei anderen Anwendungen ist dieses Vorgehen nicht praktikabel, hier wird genau ein Knopf zum Starten der Suche benötigt. Befinden sich mehrere Knöpfe mit derselben Funktionalität – möglicherweise auch noch mit unterschiedlichen Beschriftungen – auf derselben Seite, wird das von Benutzern als äußerst verwirrend empfunden.

Da die präsentierte Ergebnismenge oft sehr groß werden kann, sollte dem Benutzer die Möglichkeit gegeben werden, die Anzahl der Treffer, den dargestellten Inhalt, das Layout, die Reihenfolge oder eine Gruppierung festzulegen.

Die häufigste Unterstützung der Anfrageverfeinerung geschieht durch Benutzerbewertungen. Diese sind zwar weit verbreitet, tragen aber nicht unbedingt zur Klarheit bei, wenn der Benutzer nicht sehen kann, welche Auswirkungen die Bewertungen auf die Anfrageformulierung haben. Ein anderer Aspekt der Anfrageverfeinerung ist die Unterstützung von Folgeanfragen. Das kann zumindest dadurch geschehen, dass das System die Historie der Anfragen mitverfolgt und der Benutzer diese bei Bedarf wieder einsehen kann. Zusätzlich sollten auch die Ergebnisse abgespeichert werden können, damit sie möglicherweise in anderen Programmen (zur graphischen Visualisierung oder zur Erstellung einer Bibliographie) weiterverwendet werden können.

Abbildung 3.12 zeigt ein Beispiel einer Benutzungsschnittstelle, die den gemachten Anforderungen entspricht. Sie benötigt keine weiteren technischen Hilfsmittel, lediglich HTML-Tabellen und -Formulare.

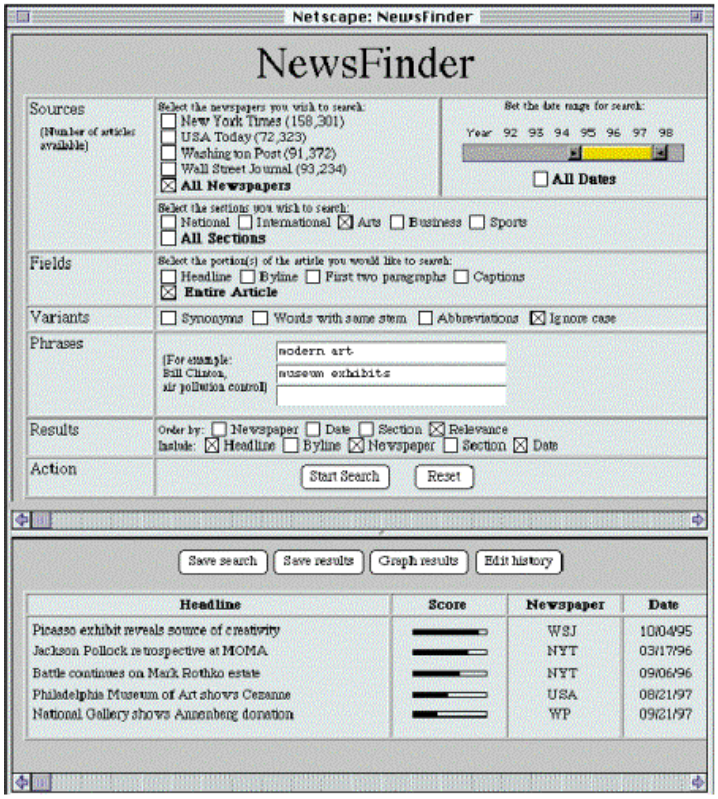


Abbildung 3.12: Beispiel für eine ideale Benutzungsschnittstelle

3.5.3 Bewertung

A1 (Einheitliche, integrierte Benutzungsschnittstelle): Eine einheitliche Benutzungsschnittstelle liegt bei allen vorgestellten Ansätzen vor. Die Benutzungsschnittstellen, die mehrere Informationsdienste integrieren, setzen entweder bereits auf einheitlichen Integrationsansätzen auf (QuakeBib, SenseMaker, DLITE) oder treten nicht mit dem Anspruch eines Meta-Recherchesystems auf (MiBiblio). Der Integrationsgrad ist allerdings bei den vorgestellten Ansätzen (mit Ausnahme von SenseMaker und dem Gestaltungsrahmen) nicht hoch, da die bibliographischen Informationen bzw. die Kurzfassungen oder Volltexte der Dokumente erst explizit vom Benutzer angefordert werden müssen und daraufhin außerhalb der gewählten Metapher angezeigt werden.

A2 (Einfache, intuitive Benutzerinteraktion): Eine einfache und intuitive Benutzerinteraktion ist im Grunde die Motivation für alle vorgestellten Ansätze. Die Benutzungsschnittstellen, die eine abstrakte Form der Visualisierung verwenden, bringen für den Benutzer zusätzliche Herausforderungen zur eigentlich beabsichtigten Informationssuche mit sich. LyberWorld wählt eine sehr abstrakte und vielschichtige Visualisierung für Dokumente in einem n-dimensionalen Raum, die sicherlich viel Abstraktionsvermögen, eine geraume

Einarbeitungszeit und ein gewisses Geschick beim Bedienen benötigt. Die Navigation innerhalb von VRML-Welten (DocumentFinder) ist keineswegs trivial und für ungeübte Benutzer nur über die Ansteuerung der vordefinierten Viewpoints handhabbar. Die Benutzerinteraktionen bei Cave-ETD bzw. QuakeBib sind möglicherweise etwas ausgereifter und angenehmer durchzuführen, stellen aber hohe Installationsanforderungen im Hinblick auf benötigte Hard- und Software. Die wenigsten Einstiegsschwierigkeiten der Benutzer werden bei den eher klassischen Ansätzen mit Desktop-Metaphern oder WIMP-Schnittstellen auftreten. SenseMaker unterstützt die Erkundung des Suchraums durch zusätzliche Strukturierungsmittel zur Sortierung und Gruppenbildung sowie durch den Einsatz von optischen Erkennungshilfen (Hi-Cites). DLITE ist besonders gut geeignet für Aufgaben, für die mehrere Bearbeitungsschritte in Folge durchgeführt werden müssen, allerdings nicht unbedingt für die Informationssuche.

A3 (Ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit): Eine ausdrucksstarke Benutzerinteraktion mit Kontrollmöglichkeit wird im Gestaltungsrahmen gezielt gefordert und an einem Anwendungsbeispiel umgesetzt. SenseMaker und QuakeBib bieten dem Benutzer die Möglichkeit, die Ergebnismenge durch Sortierungs- und Gruppierungsoperationen zu strukturieren. LyberWorld erlaubt lediglich eine individuelle Gewichtung der gewählten Suchbegriffe. MiBiblio und DLITE bieten dem Benutzer große Gestaltungsspielräume im Umgang mit den angebotenen Werkzeugen. DocumentFinder und Cave-ETD erlauben dem Benutzer hingegen keine speziellen Interaktionsmöglichkeiten mit den Dokumentenmengen.

A4 (Berücksichtigung menschlicher Eigenschaften): Zur Prozessunterstützung macht LyberWorld dem Benutzer die bereits eingeschlagenen Recherchewege bzw. die bereits betrachteten Dokumente farblich kenntlich. Eine Möglichkeit zum Abspeichern von Anfragen oder Ergebnissen bieten MiBiblio, SenseMaker, DLITE und der Gestaltungsrahmen. Eine gewisse Individualität und Bandbreite unterstützen die Ansätze, die dem Benutzer Kontroll- und Steuermöglichkeiten zur Verfügung stellen (vgl. A3). Die Ansätze mit aufwendigen graphischen Benutzungsschnittstellen stellen sicherlich zum einen zu hohe Erwartungen an die Benutzer und sind auf der anderen Seite auch nicht so schnell in der Bedienung und beim Visualisieren der Ergebnisse wie die WIMP-Ansätze. Möglicherweise sind einige Benutzer fähig, durch die ansprechende graphische Gestaltung ihre Ungeduld zu vergessen.

Zur Ausnutzung der räumlichen Orientierungsfähigkeit des Benutzers realisieren Cave-ETD und MiBiblio die Bücher an einem festen und damit für den Benutzer leicht wiederauffindbaren Platz im Raum. Bei LyberWorld, DocumentFinder und QuakeBib kann ein Buch bei verschiedenen Recherchen an unterschiedlichen Positionen auftauchen. Hier wird durch die räumliche Anordnung lediglich die Ähnlichkeitsbeziehung der Dokumente untereinander besser sichtbar gemacht.

A5 (Vollständige Anfrageplanung und -bearbeitung): Eine vollständige Anfrageplanung und -bearbeitung wird bei den meisten Ansätzen von den zugrundeliegenden Recherchesystemen durchgeführt. DocumentFinder ist allerdings auf Ergebnismengen mit maximal 200 Treffern beschränkt. Einige gewählte graphische Darstellungsarten skalieren nicht gut für große Dokumentbestände bzw. Ergebnismengen, beispielsweise werden Cave-ETD und QuakeBib dadurch deutlich schwerer zu handhaben und DLITE, MiBiblio und LyberWorld werden schnell unübersichtlich.

A6 (Duplikatbehandlung): Eine Duplikatbehandlung ist nur bei den vorgestellten Ansätzen notwendig, die mehrere Recherchedienste integrieren. SenseMaker bietet verschiedene Verfahren zur Bestimmung der Duplikate an. Bei DLITE entstehen durch die Weiterverarbeitung und Speicherung von Dokumenten Doppelungen, d.h. Redundanzen, da jedes Dokument aufgrund seiner unauflösbaren Zugehörigkeit zu einer Antwortmenge daraus nie entfernt, sondern jeweils nur vervielfältigt werden kann. Bei QuakeBib wurde bisher keine Duplikaterkennung berücksichtigt.

A7 (Schnelle Antwortzeiten): Schnelle Antwortzeiten sind nicht das Hauptziel bei der Ergebnisvisualisierung der vorgestellten räumlichen Benutzungsschnittstellen. VRML- und graphisch aufwendige Java-Anwendungen sind bekanntlich nicht die schnellsten, dafür aber Web-fähig im Gegensatz zu den eingesetzten Technologien bei LyberWorld und Cave-ETD. Die schnellsten Antwortzeiten sind nach wie vor mit klassischen WIMP-Benutzungsschnittstellen bzw. mit HTML-Technik möglich.

A8 (Berücksichtigung von Lastvorgaben): Die Berücksichtigung von Lastvorgaben der zugrundeliegenden Recherchedienste ist bei den vorgestellten prototypischen Ansätzen nicht von Bedeutung.

Resümee

Es lässt sich beobachten, dass trotz einer großen Anzahl von Prototypen für Informationsvisualisierungen und zahlreichen Forschungsprojekten die Zahl der visuellen Metaphern, die in kommerziellen Programmsystemen oder sogenannten Real-World-Applikationen im Bereich der Literaturrecherche implementiert wurden, eher gering ist. Der Grund für diese Entwicklung ist derzeit noch nicht ganz klar. Fest steht jedoch, dass viele prototypische Systeme auf dem Gebiet der Informationsvisualisierung am Benutzer vorbei entwickelt wurden. Oftmals wurden Projekte nur unter dem Aspekt der Anwendung neuer Technologien durchgeführt. Die Frage, ob und wie die vorhandenen Daten unter dem Aspekt eines Informationsmehrwerts aufbereitet und visualisiert werden sollten, wurde meist nicht behandelt.

Die Experimente der vorgestellten Ansätze zur Überprüfung der Benutzerakzeptanz liefern ebenfalls keine aussagekräftigen Anhaltspunkte und betrachten nur sehr kleine und spezielle Nutzergruppen. Für valide Ergebnisse müssen weitaus umfangreichere Studien durchgeführt werden und möglichst auch im realen Einsatzumfeld.

Die in der vorliegenden Arbeit zu entwickelnde Benutzungsschnittstelle für ein Meta-Recherchesystem hat zum Ziel, eine Entscheidungsunterstützung für den Benutzer zu liefern. Dies soll zum einen durch viel Zusatzinformation zu den gefundenen Dokumenten und zum anderen durch die direkte Vergleichbarkeit bestimmter Dokumenteigenschaften geschehen. In einem Meta-Recherchesystem ist in der Regel kein Volltextbestand verfügbar, so dass Ansätze, die die Ähnlichkeit zwischen Dokumenten aufgrund ihres Volltextes berechnen, für Meta-Suchmaschinen nicht geeignet sind.

Die vorgestellten graphischen Benutzungsschnittstellen arbeiten viel mit der Ähnlichkeit von Dokumenten. Damit begünstigen sie das Überblicken und das Analysieren der Zusammensetzung der Ergebnismenge aus einem gewissen Abstand heraus. Der Benutzer kommt allerdings meist nur durch zusätzliche Interaktionen an Titel, Autoren und gar eine Kurzfassung der Dokumente heran. In den meisten Fällen wird auch die parallele Betrachtung von Detailinformationen zu mehreren Dokumenten nicht unterstützt.

Weitaus relevanter für die vorliegende Arbeit sind die Ansätze mit Desktop-Metaphern und WIMP-Benutzungsschnittstellen. Gerade SenseMaker leistet mit den vorgeschlagenen Nachbearbeitungsoperatoren und Hi-Cites einen vielversprechenden Beitrag zur Entscheidungsunterstützung des Benutzers. Auch das Einrichten von persönlichen Bereichen für die Speicherung von Anfragen oder Dokumenten (DLITE) sollte für die vorliegende Arbeit berücksichtigt werden. Und schließlich sollten auch die konkretisierten Forderungen des Gestaltungsrahmens nach verständlichen, nachvollziehbaren, vorhersehbaren und kontrollierbaren Rechercheschnittstellen im weiteren Verlauf der Arbeit nochmals aufgegriffen werden.

3.6 Resümee

In Abschnitt 3.2 wurden die menschlichen Eigenschaften, d.h. die Fähigkeiten und Grenzen eines Benutzers erörtert und konkretisiert. Dabei wurde die große Diskrepanz zwischen den inneren Vorgängen beim kognitiven Informationsverarbeitungsprozess und den realen Suchaktionen des Benutzers deutlich. Die vorliegende Arbeit will von den realen Gegebenheiten ausgehen und versuchen, die Entwicklung des Benutzers hin zu einer umsichtigeren und mündigeren Vorgehensweise bei der Informationssuche zu fördern.

Die Betrachtung existierender Meta-Literaturrecherchedienste in Abschnitt 3.3 machte deutlich, dass der Integrationsgrad der präsentierten Suchergebnisse oft zu gering ist und damit dem Benutzer noch eine erhebliche manuelle Vergleichs- und Auswertungsarbeit aufgebürdet wird. Die Ursache dafür liegt in der Informationsportionierung der Recherchedienste im Web begründet. Daraus ergibt sich für die vorliegende Arbeit die technische Herausforderung, eine geeignete Anfragebearbeitung bzw. -optimierung zu entwickeln, die einen höheren Integrationsgrad der Ergebnisse liefern kann.

Anschließend wurden Integrationsarchitekturen betrachtet (Abschnitt 3.4), die ihren Schwerpunkt auf die Integration von heterogenen Informationsquellen legen und sich dabei in besonderem Maße auf die Anfrageplanung und -ausführung konzentrieren. Da der vorliegenden Arbeit mit der Literaturrecherche ein One-World-Szenario zugrunde liegt, ist weniger die Erstellung von korrekten und vollständigen Anfragepläne, als vielmehr die flexible und effiziente Ausführung bei der Anfragebearbeitung zu beachten. Existierende Ansätze verwenden dazu beispielsweise nicht blockierende Join-Operatoren. Für die Bearbeitung von Meta-Rechercheanfragen sind zwar keine Join-Operatoren notwendig, aber dennoch soll die Technik der nicht blockierenden Operatoren weiter verfolgt und ggf. an geeigneter Stelle eingesetzt werden.

Die vorgestellten Integrationsarchitekturen beschäftigen sich nicht mit einer geeigneten Benutzerunterstützung. Daher wurden in Abschnitt 3.5 die zwei bekanntesten Formen der Benutzerunterstützung bei der Informationssuche betrachtet: Ansätze mit Personalisierungstechniken und mit graphischen Ergebnispräsentationen.

Personalisierungsansätze sind in vielen Anwendungsbereichen nützlich, gerade wenn große Informationsmengen anfallen, die vom Benutzer zu bewältigen sind. Die inhaltsbasierten Ansätze sind allerdings zu einseitig für unser Vorhaben, sie konzentrieren sich nur auf die inhaltliche Ähnlichkeit von Textdokumenten und Anfragen. Außerdem liegen in einem Meta-Recherchesystem die Volltexte der Dokumente in der Regel nicht vor. Die kollaborativen Personalisierungsansätze lassen nur wenig Individualität zu, die nicht im Trend liegt. Geeignet sind diese Ansätze mehr für Fragen des Geschmacks im Unterhaltungsbereich als für die wissenschaftliche Literaturrecherche.

Auch wenn sich keine der beiden Personalisierungstechniken direkt für die vorliegende Arbeit anbietet, konnten doch einige Einsichten in die Beschaffenheit von Benutzerinteressen und -profilen gewonnen werden. Benutzer haben zu einem Zeitpunkt oft mehrere Interessen, Interessen können sich im Verlauf der Zeit ändern und es gibt verschiedene Qualitäten von Interessen, beispielsweise kurz- oder langfristige Interessen. Darüber hinaus sollte beim Einsatz von Benutzerprofilen dem Benutzer zum einen die Wirkweise klar sein und zum andern auch die Möglichkeit gegeben werden, sein Profil einzusehen und ggf. zu verändern.

Für die vorliegende Arbeit werden wir auf die Entwicklung einer dreidimensionalen graphischen Benutzungsschnittstelle aus den zuvor genannten Überlegungen heraus (s. Abschnitt 3.5.3) verzichten. Dafür sollen die Anregungen einiger Projekte, wie z.B. die Nachbearbeitungsoperatoren auf der Ergebnismenge, die Hi-Cites-Funktionalität bei der Ergebnispräsentation oder die Einrichtung persönlicher Arbeitsbereiche durchaus aufgegriffen und in die vorliegende Arbeit integriert werden.

4 Überblick über den Lösungsansatz

4.1 Überblick

In diesem Kapitel wird ein grober Überblick über den vorgeschlagenen Lösungsansatz gegeben. Dazu wird zu Beginn noch einmal der spezielle Handlungsbedarf identifiziert, der sich aus dem Abgleich des Anforderungskatalogs (vgl. Abschnitt 3.1) mit den bereits existierenden technischen Ansätzen und Methoden ergibt, die im vorangegangenen Kapitel eingehend betrachtet wurden (Abschnitt 4.2).

Abschnitt 4.3 benennt zunächst die Grundidee des Lösungsansatzes. Danach werden die Schwerpunkte der Arbeit ausführlicher beleuchtet und die darin enthaltenen Thesen herausgearbeitet (Abschnitt 4.4). Abschnitt 4.5 stellt die Grobarchitektur des Lösungsansatzes vor, beschreibt das Vorgehen zur Validierung des vorgeschlagenen Ansatzes sowie den zu erwartenden Beitrag der Arbeit.

Abschnitt 4.6 gibt schließlich eine Übersicht über die zur Lösung gewählte Vorgehensweise auf Kapitelebene. Die darauf folgenden Kapitel widmen sich jeweils einer detaillierten Betrachtung eines Ausschnitts bzw. Teilbereichs des Lösungsansatzes.

4.2 Handlungsbedarf

In Kapitel 2 wurde motiviert, warum Meta-Suchmaschinen ein vielversprechender Ansatz für die Unterstützung des Benutzers bei der Literaturrecherche sind. Bei der Formulierung der Anforderungen (A1 - A8) an ein solches Meta-Recherchesystem wurde deutlich, dass die Herausforderung darin besteht, einen Lösungsansatz zu finden, der den gesamten Anforderungskatalog erfüllt, da einige Anforderungen in direkter Wechselwirkung und Konkurrenz zueinander stehen. Diesbezüglich wurden zwei Spannungsfelder im Anforderungskatalog identifiziert (vgl. Abbildung 2.2). Ein Spannungsfeld betrifft die Benutzerinteraktion: Die Rechercheschnittstelle soll auf der einen Seite einfach und intuitiv zu bedienen sein, auf der anderen Seite aber auch hinreichend mächtige Ausdrucksmittel zur Verfügung stellen. Ein Lösungsansatz soll auch die menschlichen Fähigkeiten und Fertigkeiten berücksichtigen (A2, A3, A4). Das zweite Spannungsfeld betrifft die Anfragebearbeitung: Anfragen sollen möglichst vollständig integriert bearbeitet werden, die Ergebnisse sollen dem Benutzer aber auch hinreichend schnell präsentiert werden können. Dabei sollen die Duplikate eliminiert und die Lastvorgaben der eingebundenen Dienste berücksichtigt werden (A1, A5, A6, A7, A8).

In Kapitel 3 wurde daraufhin der Stand der Technik betrachtet, um bereits existierende Lösungen oder Teillösungen für die genannten Anforderungen zu ermitteln. Dabei wurden verschiedene Arten von Ansätzen untersucht.

Meta-Suchmaschinen für das WWW (Abschnitt 3.3.1) erfüllen zwar im Wesentlichen den gesamten Anforderungskatalog, allerdings liegt bei ihnen eine andere Ausgangssituation vor: Die eingebundenen Dienste betreiben keine Informationsportionierung, eine Duplikaterkennung ist basierend auf der Eindeutigkeit von URLs leicht möglich und die Ansprüche an eine Suche im Web (beispielsweise im Hinblick auch die Vollständigkeit der Ergebnisse) sind i.d.R. bedeutend geringer als an eine wissenschaftliche Literaturrecherche.

Die existierenden Meta-Recherchesysteme im Literaturbereich (Abschnitt 3.3.2) erfüllen sicherlich einige Teile des Anforderungskatalogs, liefern aber keine Lösungsansätze für das Spannungsfeld der Anfragebearbeitung. Die Systeme liefern entweder schnelle Ergebnisse, dann sind die Ergebnisse unvollständig und der Integrationsgrad ist minimal (KVK, AllBookStores). Oder die Integration der Ergebnisse erfolgt in einem etwas größeren Umfang, dann verlangsamen sich die Antwortzeiten in erheblichem Maße (Daffodil).

Die I^3-Ansätze zur Integration von heterogenen Informationsquellen legen ihr Augenmerk hauptsächlich auf den technischen Bereich der Anfragebearbeitung und vernachlässigen die Aspekte der Benutzerunterstützung. Dadurch liefern sie auch keinen Beitrag für diesen Bereich des Anforderungskatalogs (A2, A3, A4). Die Gruppe der Ansätze, die sich mit der Integration von Informationsquellen beschäftigt (Abschnitt 3.4.1), beschränkt ihre Aktivitäten im Wesentlichen auf die Erstellung von korrekten, vollständigen und minimalen Anfragepläne. Allein die Berechnung der Pläne dauert bereits mehrere Sekunden. Um die resultierende Antwortzeit des Systems abzuschätzen, muss die Zeit für die Bearbeitung der Anfrage noch hinzugerechnet werden. Die zweite Gruppe der vorgestellten Ansätze, welche sich mit der Integration von Web-basierten Informationsdiensten beschäftigt (Abschnitt 3.4.2), versucht hingegen die resultierende Anfragebearbeitungszeit zu optimieren. Dazu werden nicht blockierende Operatoren eingesetzt, welche die Streaming-Eigenschaft der Web-Dienste berücksichtigen. Die gewählten Szenarien für beide Gruppen von Ansätzen zeichnen sich allerdings durch derartig verschiedenartige Informationsquellen und -dienste aus, dass das Ziel der Zusammenführung keine vollständig integrierte Ergebnismenge ist, sondern die schnelle Verknüpfung von Ergebnissen des einen Dienstes mit Anfragen an einen darauf aufsetzenden anderen Dienst. Im Bereich der Literaturrecherche haben wir es hingegen mit einem One-World-Szenario zu tun, in welchem die Dienste im Wesentlichen dieselben Informationen zu Dokumenten bereitstellen. Die Idee der nicht blockierenden Operatoren kann aber durchaus in einem One-World-Szenario zu schneller verfügbaren Ergebnissen führen und sollte daher im Hinblick auf A7 berücksichtigt werden.

Personalisierungsansätze (Abschnitt 3.5.1) bieten Lösungen an, mit denen das Spannungsfeld des Anforderungskatalogs auf Seiten des Benutzers aufgelöst werden kann. Durch das Führen

von Benutzerprofilen kann die Anfrage des Benutzers hinreichend präzisiert werden (A3) und der Benutzer kann die Rechercheschnittstelle weiterhin in der einfachen und gewohnten Weise nutzen (A2). In einigen Fällen sind allerdings zusätzliche Eingaben des Benutzers zur Bewertung von Dokumenten bzw. Objekten notwendig (besonders bei kollaborativen Filtertechniken), teilweise wird auch von einer mangelnden Nachvollziehbarkeit für den Benutzer berichtet. Personalisierungsansätze zeichnen sich hingegen im Wesentlichen auch dadurch aus, dass sie eben keine vollständigen Ergebnisse (A5) liefern. Meist werden Personalisierungsansätze auch nur im Zusammenhang mit homogenen Informationssystemen verwendet, so dass die benötigten Informationen dort vorhanden und jederzeit zugreifbar sind. An dieser Stelle liegt genau die Schwierigkeit, die bei der Übertragung von Personalisierungstechniken auf Meta-Recherchesysteme entsteht. Die benötigten Informationen über die Dokumente oder Objekte sind in Meta-Recherchesystemen nicht (permanent) verfügbar und dürfen auch aus rechtlichen Gründen nicht zwischengespeichert werden. Ein weiterer Nachteil existierender Personalisierungstechniken liegt in der Funktionsweise der Profile. Inhaltsbasierte Benutzerprofile erweisen sich bei kurzfristigen oder sich ändernden Interessen als schwierig und sind nur für die Unterstützung von längerfristigen Informationsbedürfnissen sinnvoll. Kollaborative Techniken eignen sich gut für Geschmacksfragen auf unterschiedlichen Arten von Objekten, fördern allerdings die Meinungen von Mehrheiten. Für die Unterstützung der wissenschaftlichen Literaturrecherche sind kollaborative Techniken somit nicht geeignet.

Graphisch aufbereitete Rechercheschnittstellen (Abschnitt 3.5.2) bieten Lösungen an, die eine intuitive Benutzerinteraktion ermöglichen und die menschliche Fähigkeiten und Fertigkeiten in besonderem Maße berücksichtigen. Die meisten räumlichen Ergebnisdarstellungen beziehen sich allerdings – wie die Personalisierungsansätze auch – auf ein homogenes Informationssystem. Dadurch sind genügend Informationen zu den einzelnen Dokumenten verfügbar, die für eine entsprechende Visualisierung verwendet werden können. Diese Ansätze lassen sich daher ebenfalls nur schwer auf Meta-Recherchesysteme übertragen. Weiterhin kommt hinzu, dass einige der vorgestellten Ansätze nicht über eine Web-schnittstelle angeboten werden können (z.B. LyberWorld, Cave-ETD). Die Gruppe der textbasierten WIMP-Ansätze bietet ebenso geeignete Lösungsansätze für das Spannungsfeld der Benutzerinteraktion an. Die vorgeschlagenen Techniken lassen sich auch im Rahmen von Web-basierten Meta-Recherchesystemen einsetzen. Die Ergebnisse werden kontinuierlich präsentiert, so dass der Benutzer keine langen Wartezeiten bemerkt. Allerdings verfolgen diese Ansätze keine vollständige Ergebnisintegration (SenseMaker, DLITE) und können nicht zur Lösung des zweiten Spannungsfeldes des Anforderungskatalogs beitragen.

Zusammenfassend ergibt sich daraus, dass zwar Teile des Anforderungskatalogs von existierenden Ansätzen gelöst werden können, aber bisher keine Lösungen für das Spannungsfeld der Anfragebearbeitung zur Verfügung stehen (A1, A5, A6, A7, A8). Existierende Ansätze priorisieren i.d.R. schnelle Antwortzeiten (A7) und nehmen dafür unvollständige Ergebnismengen (A5) und eine erhebliche manuelle Integrationsarbeit des Benutzers (A1, A5, A6) in Kauf.

Der Handlungsbedarf für die vorliegende Arbeit besteht daher größtenteils in der Bereitstellung von technischen Lösungen für eine geeignete Anfragebearbeitung, darf aber die genannten Benutzungsaspekte dabei nicht vernachlässigen. Dies kann beispielsweise dadurch geschehen, dass Ideen existierender Lösungsansätze verwendet und für die vorliegende Arbeit in geeigneter Weise angepasst werden.

Die zentrale Herausforderung des benötigten Lösungsansatzes liegt in einer möglichst vollständigen Beschaffung und Integration der Ergebnisse und gleichzeitig in einer möglichst schnellen Präsentation der Ergebnisse. Diese Problematik ergibt sich im Wesentlichen aus der Informationsportionierung der eingebundenen Dienste. Da die Literaturdienste ihre Ergebnisse nur portionsweise zur Verfügung stellen, müssen all diese Informationsportionen erst aufgesammelt und miteinander kombiniert werden. Dies ist von zentraler Bedeutung für die vorliegende Arbeit. Daher wird die vorläufige Definition 3.1 der Informationsportionierung (S. 57) im Folgenden nochmals aufgegriffen und in technischer Hinsicht verfeinert.

Web-basierte Literaturdienste präsentieren die Ergebnisse einer Anfrage nicht zusammenhängend, sondern verteilen die Informationen zu den Ergebnissen auf mehrere Portionen. Eine Informationsportion wird jeweils auf einer HTML-Seite präsentiert.

Definition 4.1: Werden zu einer Anfrage nicht alle Ergebnisse (zumindest mit ihren Kurztiteln) auf einer HTML-Seite präsentiert, sprechen wir von einer *horizontalen Informationsportionierung*. Der Benutzer muss beispielsweise über eine „Mehr Treffer"-Schaltfläche explizit eine zusätzliche Informationsportion anfordern, die dann weitere Ergebnisse (mit ihren Kurztiteln) beinhaltet.

Definition 4.2: Werden zu einem Ergebnistreffer nicht alle verfügbaren Informationen präsentiert, sprechen wir von einer *vertikalen Informationsportionierung*. Der Benutzer muss beispielsweise über die Repräsentation eines Treffers als Kurztitel bzw. eine „Detail"-Schaltfläche eine weitere Informationsportion anfordern, welche dann sämtliche oder zumindest zusätzliche Informationen zu einem gefundenen Dokument beinhaltet.

In der Regel verwenden Literaturdienste sowohl horizontale als auch vertikale Informationsportionierungen. Üblich sind beispielsweise (horizontale) Informationsportionen mit 10, 30, 50 oder 100 Kurztiteln und zwei bis drei (vertikale) Informationsportionen zu jedem Ergebnisdokument.

Meta-Recherchesysteme binden existierendende Literaturdienste über deren HTTP-Schnittstelle ein. Um eine Informationsportion anzufordern, ist also ein HTTP-Aufruf notwendig. Um das gesamte Ergebnis einer Anfrage anzufordern, sind entsprechend viele HTTP-Aufrufe notwendig, nämlich für jede benötigte Informationsportion ein HTTP-Aufruf. HTTP-Aufrufe benötigen zum einen eine gewisse Zeit (diese liegt im Bereich von einigen Sekunden), zum anderen erzeugen sie auf den Literaturdiensten eine nicht unerhebliche Last. Daher können nicht beliebig viele HTTP-Aufrufe gleichzeitig abgesetzt werden, was dazu führt, dass die Bereitstellung einer vollständigen und integrierten Ergebnismenge eine gewisse Zeit benötigt.

Die zentrale Herausforderung für den gesuchten Lösungsansatz besteht also darin, mit der anfallenden Menge an Informationsportionen, d.h. an HTTP-Aufrufen, so umzugehen, dass die verfügbaren Informationen möglichst vollständig und möglichst schnell zusammengetragen werden können.

4.3 Grundidee des Lösungsansatzes

Um den Bereich der Benutzerinteraktion in geeigneter Weise zu unterstützen, wird die Idee der Benutzerprofile aufgegriffen. Durch die zusätzlichen Angaben in Benutzerprofilen können Anfragen präziser formuliert werden, so dass damit einer Informationsüberflutung des Benutzers entgegengewirkt werden kann. Dabei werden keine allzu hohen Erwartungen an den Benutzer gestellt (A4). Er kann weiterhin einfache Anfragen verwenden, die durch die Informationen der Benutzerprofile zu aussagekräftigen Anfrageformulierungen werden können (A2, A3). Allerdings werden beim vorgeschlagenen Lösungsansatz in den Benutzerprofilen weder inhaltliche Vorlieben vermerkt noch werden mit kollaborativen Techniken Ähnlichkeiten zwischen Benutzern ermittelt. Die Benutzerprofile des vorgeschlagenen Lösungsansatzes spezifizieren die Informationen, die für den Benutzer wichtig sind, um seine Entscheidungen treffen zu können.

Die zentrale Idee des Lösungsansatzes besteht darin, die zur Literaturrecherche bereitgestellte Anfragesprache um weiche Bedingungen, sogenannte Präferenzen, zu erweitern. Mit weichen Bedingungen kann der Benutzer beispielsweise „lieber als“-Bedingungen formulieren. Bei der Literaturrecherche wurde der Einfluss von Benutzervorlieben bereits in zahlreichen Studien bemerkt und als bedeutsam identifiziert. Vorlieben entsprechen zum einen der intuitiven Vorgehensweise bei der Literaturrecherche (A2) und sind gut geeignet, um die Informationsbedürfnisse der Benutzer präziser auszudrücken (A3). Zum anderen liefern derartige Präferenzen für die Anfragebearbeitung auch die entscheidenden Anhaltspunkte, um die gleichzeitige Forderung nach Vollständigkeit (A1, A5) und Schnelligkeit (A7) der Ergebnispräsentation erfüllen zu können. Durch die Spezifizierung von Präferenzen können gerade die bevorzugten Ergebnisse besonders schnell beschafft werden, ohne damit prinzipiell die Vollständigkeit der Ergebnismenge zu beeinträchtigen.

In Integrationsarchitekturen verdecken Wandler i.d.R. nicht nur die semantische Heterogenität der eingebundenen Dienste, sondern auch die technische Heterogenität, die beispielsweise auch die Informationsportionierung der eingebundenen Dienste mit einschließt. Für die geforderte Anfragebearbeitung ist es entscheidend, die Funktionalität der Wandler, die den Umgang mit den Informationsportionen betrifft, detaillierter zu kennen und direkt beeinflussen zu können. Ein Blackbox-Ansatz, wie er von den meisten Integrationsansätzen für die Umsetzung der Wandler gewählt wird, kann also für den gesuchten Lösungsansatz nicht verwendet werden. Daher werden der Anfragebearbeitung im vorgeschlagenen Lösungsansatz die notwendigen Informationen bzgl. der Informationsportionierungen und der Lastvorgaben der eingebundenen Dienste zur Verfügung gestellt (A1, A5, A7, A8).

Durch die genannten Ideen können die kritischen Anforderungen, d.h. das Spannungsfeld der Benutzerinteraktion sowie das Spannungsfeld der Anfragebearbeitung, gleichzeitig erfüllt werden. Um den gesamten Anforderungskatalog abzudecken, sind flankierend dazu weitere Konzepte notwendig, die die Einheitlichkeit der Ergebnispräsentation durchsetzen (A1) und eine geeignete Duplikatbehandlung vornehmen können (A6).

4.4 Schwerpunkte der Arbeit

4.4.1 Benutzerprofile

Die vorliegende Arbeit argumentiert, dass durch den Einsatz von Benutzerprofilen präzisere Anfragen möglich werden. Präzise Anfragen sind für eine effektive Anfragebearbeitung und -optimierung entscheidend. Darüber hinaus muss der Benutzer nicht allzu viel Zeit und Mühe bei der aktuellen Anfrageformulierung investieren, sofern geeignete Benutzerprofile vorhanden sind.

Die vorgeschlagenen Benutzerprofile werden *generische Suchmuster* genannt. Ein generisches Suchmuster enthält in der Regel keine thematischen Suchbegriffe, sondern Wünsche und Vorlieben, die die anderen Eigenschaften von Dokumenten betreffen, beispielsweise Sprache, Beschaffungsdauer oder Beschaffungskosten. Benutzerstudien (vgl. Abschnitt 3.2) haben gezeigt, dass das Suchverhalten eines Benutzers am stärksten von seinen eigenen Vorlieben und den vorhandenen Fertigkeiten und Kenntnissen bei der Informationssuche geprägt ist. Bei der Entscheidungsfindung des Benutzers ist die thematische Eignung des Dokuments meist nur ein Kriterium unter vielen, wobei andere Aspekte wie Aktualität, renommierter Autor, gute Leserkritiken oder eine rasche Verfügbarkeit oft sogar wichtiger sein können.

Die vorgeschlagene Form der Personalisierung fällt also weder in die Klasse der inhaltsbasierten noch in die Klasse der kollaborativen Personalisierungstechniken. Am treffendsten könnte man sie vielleicht als Nutzwert-basierte (*utility-based*) Personalisierung bezeichnen [Bur02]. Der Benutzer gibt die Eigenschaften von Dokumenten an, die für ihn den

größten Nutzwert haben und auf deren Basis er seine Entscheidung voraussichtlich treffen wird. Im folgenden Abschnitt wird diese Konzeption weiter präzisiert werden (vgl. Definition 4.3).

These 1: Der Einsatz von Benutzerprofilen (in der vorliegenden Arbeit von generischen Suchmustern) kann Benutzeranfragen präzisieren. Dadurch können die Anfragen von einem vollständig integrierten Meta-Recherchesystem schneller beantwortet werden.

Im Rahmen des vorgeschlagenen Lösungsansatzes werden Benutzerprofile nicht automatisch erzeugt, da diese Vorgehensweise weder zur Klarheit und Nachvollziehbarkeit noch zu Lernerfahrungen des Benutzers beiträgt. Die Benutzerprofile werden also – wenn möglich – vom Benutzer selbst erstellt (mit einer geeigneten Unterstützung) und können dabei, aber auch im laufenden Betrieb, eingesehen und verändert werden.

Jeder Benutzer erhält ein eigenes Repertoire an generischen Suchmustern, die er bei Bedarf als Ergänzung einer thematischen Suchanfrage verwenden kann. Beispielsweise kann ein Suchmuster für „sofort ausleihbare Lehrbücher in Bibliotheken vor Ort“ formuliert werden, welches der Benutzer dann verwenden kann, wenn er ein Buch zum Thema „Java“ oder zum Thema „Datenbanken“ benötigt. Das Repertoire an Suchmustern kann sich im Laufe der Zeit gemäß der Vorlieben oder Lernerfahrungen des Benutzers verändern, es kann beispielsweise umfangreicher oder ausgefeilter werden.

4.4.2 Anfragebearbeitung und -optimierung

Die Anfrageplanung bei Meta-Recherchesystemen im Bereich der Literaturrecherche und -beschaffung kann relativ gradlinig erfolgen. Eine gegebene Benutzeranfrage kann meist ohne Umstrukturierungsmaßnahmen an die zugrundliegenden Informationsdienste weitergeleitet werden, da die Ausdrucksmächtigkeit der angebotenen Anfragesprachen meist identisch ist. Die tatsächliche Anfragetransformation vom internen Format in die geforderte Syntax und Semantik der Dienste übernehmen die Wandler.

Das weitaus größere Problem bei der Anfragebearbeitung stellt nun die Lieferung der Ergebnisse dar. Die unterschiedliche Semantik der Ergebnisse kann wiederum von den Wandlern bereinigt werden. Da die Dienste ihre Ergebnisse in Form von Informationsportionen bereitstellen, die über einzelne HTTP-Aufrufe aufgesammelt werden müssen, können die Wandler zwar die einzelnen Aufrufe verdecken, nicht aber die dadurch entstehenden Antwortzeiten. Für eine effektive Anfragebearbeitung ist der Umgang mit diesen Informationsportionen allerdings von zentraler Bedeutung. Daher betrachten wir in der vorliegenden Arbeit Wandler in erster Linie als Transformatoren zwischen unterschiedlichen Anfrage- und Ergebnisformaten, die sich an den Informationsstrukturen der Dienste orientieren und keine eigenständigen Anfrageplanungen bzw. -optimierungen ausführen. D.h. die Menge und Reihenfolge der zu beschaffenden Informationsportionen kann explizit von der Anfrageplanung gesteuert werden.

Durch die eingeführten Benutzerprofile sind präzisere Anfragen möglich, d.h. Anfragen, die mehrere Angaben beinhalten. Zusätzlich wird die Anfragesprache um weiche Bedingungen (Präferenzen) erweitert. Der Benutzer kann also neben den harten Bedingungen wie „Sprache=deutsch“ oder „Jahr=2003“ auch Präferenzen angeben, beispielsweise „Sprache: lieber deutsch als englisch“ oder „Jahr: möglichst aktuell“. Bei Benutzerstudien hat sich herausgestellt, dass Benutzer oft deshalb weniger Angaben bei einer Anfrage machen, weil sie befürchten, dass ihnen dadurch relevante Dokumente verloren gehen. Die Formulierung von Präferenzen hingegen ändert die Zusammensetzung der Ergebnismenge nicht, sondern beeinflusst lediglich die Gewichtung bzw. die Reihenfolge der Dokumente innerhalb der Ergebnismenge.

Benutzerstudien haben gezeigt, dass die Relevanz eines Dokuments für einen Benutzer von den unterschiedlichsten Kriterien abhängen kann (s. Abschnitt 4.3.1). Daher betrifft der hier verwendete Relevanzbegriff nicht nur die thematische Ausrichtung von Dokumenten. Relevanz bedeutet vielmehr: Ein Benutzer bekommt zu einem Dokument die Informationen angezeigt, die für seine Entscheidungsfindung hilfreich sind und damit einen sog. Nutzwert haben.

Definition 4.3: Ein Dokument ist *(entscheidungs-)relevant*, wenn es Angaben zu einem Merkmal enthält, das ein Benutzer für seine Entscheidungen heranzieht.

Ein häufig genanntes Entscheidungskriterium (Merkmal) ist die Beschaffungsinformation, d.h. wann ein bestimmtes Dokument für den Benutzer verfügbar ist (Lieferzeit) und was die Beschaffung für den Benutzer kostet (Lieferkosten). Gerade die Lieferzeiten können aus Bibliothekskatalogen aufgrund der dreistufigen vertikalen Informationsportionierung nur mit erheblichem Aufwand bezogen werden. Buchhändler hingegen haben die Bedeutung dieser Information bereits erkannt, stellen sie zu jedem Dokument auch sofort zur Verfügung und erlauben sogar eine Ergebnissortierung anhand von Lieferzeiten.

Beim vorliegenden Relevanzbegriff gibt es folglich noch eine Abstufung: Ein Dokument, zu dem keine Angaben über Lieferzeiten gemacht werden können, ist nicht relevant. Ein Dokument, welches Lieferzeiten angeben kann, auch wenn diese für den Benutzer nicht gerade erfreulich sind, ist relevant. Und ein Dokument, welches eine Lieferzeit angibt, und noch dazu eine sehr schnelle, ist besonders relevant.

Um die Antwortzeiten für den Benutzer so kurz wie möglich zu gestalten, erfolgt der Ergebnismengenaufbau kontinuierlich. Die Ergebnisse von den eingebundenen Diensten werden mit möglichst geringen Verzögerungen – beispielsweise im Falle einer Duplikatbehandlung – an den Benutzer weitergereicht. Bei existierenden Systemen ist das Anwachsen bzw. die Reihenfolge der Ergebnismenge hauptsächlich von den reinen Antwortzeiten geprägt, d.h. die Ergebnisse der schnellsten Dienste erscheinen zuerst in der Ergebnismenge.

In Benutzerstudien wurde aber auch deutlich, dass Benutzer gerade den ersten Ergebnissen besondere Aufmerksamkeit schenken. Die vorliegende Arbeit verfolgt daher den Ansatz, durch die Unterstützung von Präferenzen nicht nur die schnellsten Ergebnisse, sondern gleichzeitig auch die relevantesten Ergebnisse zuerst zu präsentieren.

These 2: Mithilfe von weichen Bedingungen, sog. Präferenzen, kann die Anfragebearbeitung die Forderungen nach Vollständigkeit und Schnelligkeit der Ergebnispräsentation miteinander vereinbaren und somit bestmöglich erfüllen.

Das angestrebte Optimierungsziel beinhaltet also zwei Aspekte: Zum einen sollen die Ergebnisse möglichst schnell geliefert werden, und zum anderen wird eine vollständige Ergebnislieferung erwartet, wobei die Lieferung von relevanten Dokumenten der Lieferung von weniger relevanten Dokumenten vorgezogen wird. Damit wird die Forderung nach Vollständigkeit entsprechend der Relevanz der Dokumente aufgeweicht bzw. differenziert betrachtet. Durch diesen Ansatz kann die scheinbare Widersprüchlichkeit der Forderungen nach Vollständigkeit und schnellen Antwortzeiten relativiert werden. In die Kostenfunktion für die Optimierung fließt also sowohl die Antwortzeit als auch die Relevanz der Ergebnisse ein.

Der Optimierungsbegriff ist für die vorliegende Arbeit weiter zu präzisieren: Bei Optimierungsfragen ist generell zu unterscheiden, ob eine einzelne Benutzeranfrage optimiert werden soll (*local query optimization*), die Menge der gleichzeitig aktiven Benutzeranfragen oder eine Folge von Benutzeranfragen (*global query optimization*). Die bisher angestellten und folgenden Betrachtungen beziehen sich auf die lokale Optimierung einer einzelnen Benutzeranfrage.

4.5 Lösungsansatz

Der vorgeschlagene Lösungsansatz geht von der in Abschnitt 2.4 vorgestellten UniCats-Gesamtarchitektur (s. Abbildung 2.1) für ein Meta-Recherchesystem im Bereich der wissenschaftlichen Literaturversorgung aus und verfeinert dazu die Komponente des Benutzerassistenten.

4.5.1 Grobarchitektur des Benutzerassistenten

Abbildung 4.1 gibt einen Überblick über die Konzeption und die wesentlichen Module des Benutzerassistenten sowie über das Zusammenspiel mit einem Trader (optional) und den notwendigen Wandlern.

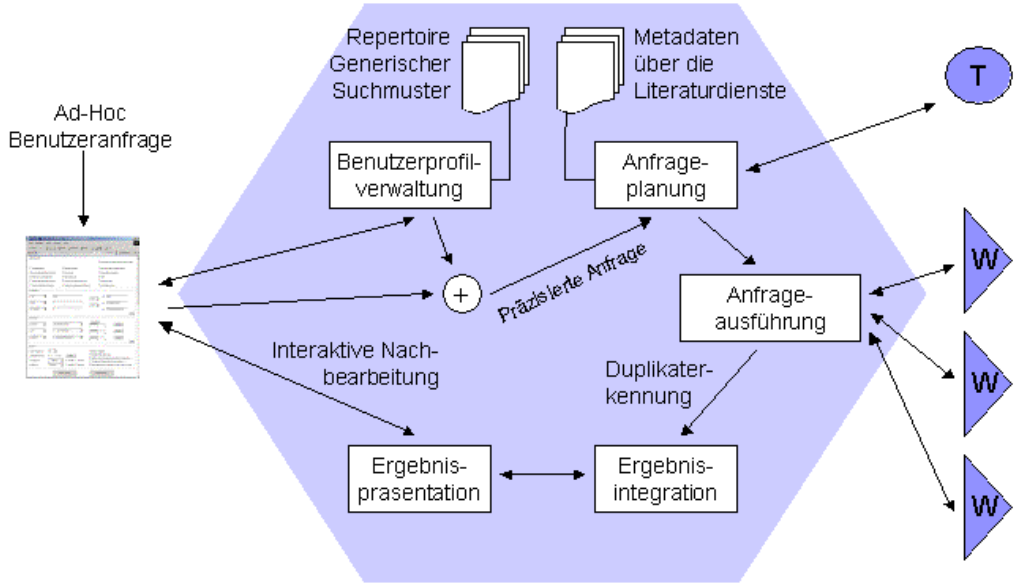


Abbildung 4.1: Grobarchitektur des Benutzerassistenten im Zusammenspiel mit Trader (T) und Wandlern (W)

Zu Beginn einer Recherche wird vom Benutzer eine „einfache“ Suchanfrage erwartet, die nur wenige thematische Suchterme beinhalten muss. Diese ad-hoc-Anfrage kann mit einem gespeicherten generischen Suchmuster kombiniert werden, welches Zusatzangaben zu den Wünschen und Vorlieben des Benutzers enthält. Diese Wünsche bzw. Vorlieben können beispielsweise die Sprache, den gewünschten Publikationstyp oder die Beschaffungsmodalitäten der gesuchten Dokumente näher beschreiben. Der Benutzer kann sein eigenes Repertoire an generischen Suchmustern jederzeit einsehen und verändern. Die Verknüpfung der ad-hoc-Anfrage mit einem generischen Suchmuster bildet die präzisierte Anfrage, die an die Komponente der Anfrageplanung weitergeleitet wird.

Die Anfrageplanung besitzt nun Metadaten über die eingebundenen Dienste, beispielsweise über die Informationsportionierung oder über Lastbeschränkungen, und kann damit einen geeigneten Anfrageplan erstellen. Im Wesentlichen werden die gewünschten Dienste parallel befragt. Die Anfrageplanung legt die Menge und die Reihenfolge der zu beschaffenden Informationsportionen fest. Für die Wahl der Reihenfolge und die Bestimmung der Menge der notwendigen HTTP-Aufrufe können eine Reihe von verschiedenen Verfahren angegeben werden. Falls die anzufragenden Dienste nicht explizit vom Benutzer ausgewählt wurden, kann der Trader über passende Dienste Auskunft geben.

Die Anfrageausführung ist für die Durchführung der notwendigen HTTP-Aufrufe verantwortlich. Dabei hat sie die Aufgabe, die vorgegebenen Lastbeschränkungen der befragten Dienste zu berücksichtigen und einzuhalten. Die HTTP-Aufrufe werden nicht direkt auf den zugrundeliegenden Diensten ausgeführt, sondern an die entsprechenden Wandler

weitergeleitet, welche die Transformationen zwischen dem einheitlichen internen und den unterschiedlichen nativen Anfrage- und Ergebnisformaten der Dienste durchführen.

Die Komponente der Ergebnisintegration sammelt die eintreffenden Ergebnisse der HTTP-Aufrufe in einer Ergebnismenge auf. Diese Ergebnismenge wächst kontinuierlich in der Weise wie neue Ergebnisse eintreffen. Beim Einsortieren der neu eintreffenden Ergebnisse ist die Behandlung von Duplikaten vorzusehen. Dazu werden geeignete Algorithmen zum Erkennen und zum Verschmelzen von Duplikaten benötigt, die zudem den Datenfluss möglichst nicht blockieren sollen.

Diese kontinuierlich wachsende Ergebnismenge wird von einer Präsentationskomponente für den Benutzer aufbereitet. Dabei kann der Aktualisierungstakt reguliert werden, falls die Aktualisierung an der Benutzungsschnittstelle beispielsweise nur alle 10 Sekunden erfolgen soll. Auf der Ergebnismenge werden interaktive Nachbearbeitungsoperationen angeboten, durch die der Benutzer die Sortierung und die Gruppierung der Treffer verändern kann. Die Präsentationskomponente beeinflusst daher auch die Ergebnisintegration, da diese Komponente die Operatoren zum Sortieren und Gruppieren der Ergebnismenge beinhaltet. Die Präsentationskomponente bietet eine geeignete graphische Darstellung für die Entscheidungsunterstützung des Benutzers bei der Betrachtung der Ergebnisse. Prinzipiell ist es auch möglich, verschiedene Arten der Ergebnispräsentation parallel für das Meta-Recherchesystem anzubieten, so dass der Benutzer die für ihn am besten geeignete Darstellung wählen kann.

4.5.2 Validierung

Im vorangehenden Abschnitt wurde bereits angedeutet, dass bei der Komponente der Anfrageplanung und der Vorgehensweise bei der Duplikaterkennung einige Gestaltungsspielräume existieren, die mit geeigneten Verfahren und Strategien besetzt werden können. Hier stellt sich die Frage, wie festgestellt werden kann, welche Verfahren geeigneter sind als andere bzw. unter welchen Bedingungen welche Verfahren geeigneter sind als andere.

Zur Validierung ist es nicht ausreichend, nur theoretische bzw. konzeptuelle Betrachtungen anzustellen. Eine Überprüfung basierend auf realen Diensten ist allerdings auch weder durchführbar (wegen bestehender Lastvorgaben) noch hinreichend umfassend, da auf diese Weise nur die Ist-Situation, was Antwortzeiten und Informationsportionierungen der Literaturdienste betrifft, berücksichtigt werden kann.

Daher soll zur Validierung der vorgeschlagenen Verfahren ein Simulationsansatz verwendet werden, welcher die zugrundeliegenden Dienste in ihrem strukturellen und zeitlichen Verhalten nachbilden kann. Damit können verschiedene Verfahren zur Anfrageplanung bzw. zur Duplikaterkennung oder auch Kombinationen von diesen Verfahren in unterschiedlichen Szenarien getestet werden. Beispielsweise können Szenarien mit nur wenigen ähnlichen

Diensten, mit vielen gleich strukturierten Diensten oder mit zeitlich sehr unterschiedlich antwortenden Diensten evaluiert werden.

Für eine Validierung wird ein Maß zur Vergleichbarkeit und zur Bewertung der eingesetzten Verfahren benötigt. Ein solches Maß entspricht im Wesentlichen dem in Abschnitt 4.4.2 formulierten Optimierungsziel und der benötigten Kostenfunktion. Das Bewertungsmaß wird sich auf den kontinuierlichen Aufbau der Ergebnismenge beziehen und beobachten, wie schnell die Ergebnisse eintreffen und wie relevant die eintreffenden Ergebnisse sind.

4.5.3 Beitrag

Zur Beantwortung der in Abschnitt 4.4 formulierten Thesen sind qualitative und quantitative Aussagen notwendig. Durch den Simulationsansatz sollen diese Thesen quantitativ belegt werden.

Im Bereich der Benutzerprofile und der Personalisierung existieren viele Arbeiten, die sich damit beschäftigen, welche Daten vom Benutzer erhoben werden können und dürfen, welches die besten Verfahren zum automatischen Erheben dieser Daten sind und wie diese Daten in das Systemverhalten einfließen können. Die vorliegende Arbeit wählt einen anderen Ansatz: Hier soll die Wirksamkeit von einzelnen Benutzerangaben in einem Benutzerprofil untersucht werden, im Hinblick auf eine gezieltere Anfragebearbeitung und damit auf eine Vermeidung der Informationsüberflutung. In einem Benutzerprofil können die verschiedensten Merkmale und Eigenschaften von Dokumenten gefordert (harte Bedingungen) bzw. bevorzugt (weiche Bedingungen) werden.

Der Simulationsansatz soll dazu beitragen, verschiedenartige Merkmale und Bedingungen in Benutzerprofilen daraufhin zu überprüfen, welche Selektivitätseigenschaften sie aufweisen und wie dadurch die Performanz der Anfragebearbeitung beeinflusst werden kann. Dabei sollen verschiedene Merkmale und Bedingungen nicht nur alleinstehend, sondern auch in Kombinationen untersucht werden. Performanz soll im Folgenden nicht nur eindimensional im Hinblick auf schnelle Ergebnisse verstanden werden, sondern bezieht sich auf die Erfüllung des zuvor formulierten Optimierungskriteriums, welches eben die Vollständigkeit und die Schnelligkeit der Ergebnispräsentation gleichermaßen berücksichtigt.

Es ist bekannt, dass Benutzer nur ungern persönliche Daten preisgeben oder zu bequem bzw. zu ungeduldig zum expliziten Eingeben sind. Mit dem Simulationsansatz soll untersucht werden, wie viele Angaben mindestens und höchstens notwendig sind, um eine deutlich bessere Performanz der Anfragebearbeitung zu erreichen. Dabei liegt die Vermutung nahe, dass nicht immer „mehr“ Bedingungen bzw. „mehr“ Präferenzen auch „bessere“ Ergebnisse liefern. Es ist durchaus denkbar, dass bereits mit Angaben von einigen wenigen Merkmalen und Bedingungen die kritische Menge an zusätzlichen Hinweisen erreicht wird, die zu akzeptablen Antwortzeiten führt.

Neben ausführlichen quantitativen Betrachtungen der genannten Thesen besteht der Beitrag der vorliegenden Arbeit in der Entwicklung eines Rahmenwerks für Meta-Recherchesysteme im Bereich der wissenschaftlichen Literaturversorgung. Durch den Simulationsansatz können verschiedene Algorithmen und Techniken zur Anfragebearbeitung untersucht und miteinander verglichen werden, beispielsweise im Hinblick auf eine geeignete Koordinierung der HTTP-Aufrufe oder auf geeignete Duplikaterkennungsstrategien. Durch die Bewertung dieser Vorgehensweisen in unterschiedlichen Szenarien können Aussagen darüber getroffen werden, welche Verfahren und Techniken zum Standardrepertoire eines Meta-Recherchesystems gehören sollten und bei welcher Art von Anfrage welche Verfahren und Techniken eingesetzt werden sollten.

4.6 Weiteres Vorgehen

Die nun folgenden Kapitel gehen detaillierter auf den vorgeschlagenen Lösungsansatz ein.

Innerhalb des Benutzerassistenten wird ein gemeinsames Informationsmodell für die Repräsentation der Dokumente verwendet (A1). Dieses Informationsmodell definiert einen Satz an Beschreibungsmerkmalen, welche in Benutzeranfragen oder in generischen Suchmustern näher charakterisiert werden können. Das Informationsmodell wird in Kapitel 5 entwickelt. Kapitel 6 führt die intern verwendete Anfragesprache ein, welche hinreichend ausdrucksmächtig sein soll und dazu die Verwendung von harten und weichen Bedingungen, also Präferenzen, unterstützt (A3). Kapitel 7 geht detaillierter auf das Konzept der generischen Suchmuster ein, welches zur Reduktion der Anfragekomplexität und zur Unterstützung einfacher und intuitiver Benutzeranfragen dient (A2, A4). In Kapitel 7 wird zudem ein Gestaltungsvorschlag ausgearbeitet, der die entwickelten Konzepte in geeigneter Weise an der Benutzungsschnittstelle umsetzt (A4).

Kapitel 8 schafft die Grundlagen für die Anfragebearbeitung. Hier werden existierende Literaturdienste genauer in ihren Anfragefunktionalitäten, ihren Informationsportionierungen und in ihrem Antwortzeitverhalten untersucht. Daraus kann ein geeignetes Datenformat für die Metadaten gewonnen werden, welche zur Anfrageplanung berücksichtigt werden müssen. Die Anfrageplanung und -ausführung beschäftigt sich mit verschiedenen technischen Strategien zur Koordinierung der notwendigen HTTP-Aufrufe. Dazu werden in Kapitel 9 verschiedene Verfahren und Algorithmen entwickelt, die zudem gegebene Lastvorgaben von Seiten der Literaturdienste einhalten können (A8). Auf den eintreffenden Ergebnissen findet nun die Duplikaterkennung und -eliminierung statt. Kapitel 10 erörtert zunächst verschiedene Definitionsmöglichkeiten für das Konzept der „Duplikate“ und passt einige Algorithmen aus dem Bereich der toleranten Zeichenkettenvergleiche für die Verwendung im Literaturszenario an (A6). Auch theoretische Betrachtungen der algorithmischen Komplexität werden dabei angestellt. Kapitel 11 beschäftigt sich mit der Entwicklung einer geeigneten Kostenfunktion für die Bewertung der Performanz der gesamten Anfragebearbeitung (A5, A7). Dazu werden

einige Bewertungsmaße entwickelt, die nicht nur technischen Gegebenheiten, sondern auch Forderungen von Seiten der Benutzer Rechnung tragen.

Zur Validierung des Lösungsansatzes wird in Kapitel 12 eine Simulationsumgebung entwickelt, die die Durchführung der Experimente in geeigneter Weise unterstützt. Die Literaturdienste werden dabei in ihren strukturellen Eigenschaften und in ihrem Antwortzeitverhalten nachgebildet, um dadurch die entwickelten Strategien zur Anfragebearbeitung sowohl in realen als auch in beliebigen Szenarien untersuchen und bewerten zu können.

Kapitel 13 berichtet über die durchgeführten Experimente und formuliert daraus eine Reihe von Ergebnissen und Erkenntnissen. Anhand der Ergebnisse können die Thesen der Arbeit quantitativ untermauert werden. Aus den gewonnenen Ergebnissen und Erkenntnissen werden schließlich Handlungsempfehlungen für die Entwicklung von Meta-Recherchesystemen und Literaturdiensten abgeleitet.

5 Domänenmodell

5.1 Überblick

Nach einer kurzen Motivation, in welcher die zwei gängigsten Vorgehensweisen bei der Erstellung eines einheitlichen Informationsmodells vorgestellt werden (Abschnitt 5.2), werden die Anforderungen an das für den vorliegenden Ansatz zu erstellende Domänenmodell formuliert (Abschnitt 5.3). Anhand von Benutzerstudien und Betrachtungen existierender Recherchedienste wird die geforderte Abdeckungsbreite und der angemessene Abstraktionsgrad des Domänenmodells weiter konkretisiert. Anschließend werden bekannte Beschreibungsmodelle für Literaturdaten vorgestellt und dahingehend untersucht, ob sie sich für die Verwendung in unserem Domänenmodell eignen (Abschnitt 5.4). Schließlich wird ein eigenes Domänenmodell entworfen (Abschnitt 5.5). Dazu werden die verwendeten Konzepte erläutert und in Form einer DTD formalisiert.

5.2 Grundsätzliche Vorgehensweisen bei der Modellerstellung

Bei der Erstellung eines einheitlichen Informationsmodells für heterogene Informationsquellen können zwei grundsätzliche Vorgehensweisen unterschieden werden [BKLW99]: die Erstellung eines globalen Schemas oder eines Domänenmodells.

Der erste Ansatz bei der Integration von heterogenen Informationsquellen ist die Erstellung eines globalen Schemas, welches sämtliche Informationen der zugrundeliegenden Quellen enthält und gegen welches dann die Anfragen gestellt werden. Die Entwicklung eines globalen Schemas kann man auch als Bottom-Up-Ansatz sehen, in welchem die auftretenden Integrationskonflikte zwischen den einzelnen Schemata schrittweise bereinigt werden. Verschiedene Techniken sind dazu möglich, z.B. die zusicherungsbasierte Integration, die formalisierte objektorientierte Integration oder die Integration über ein generisches Integrationsmodell. Einen guten Überblick liefert [Con97]. Meist wird dieser Ansatz gewählt, wenn klassische Datenbankmodelle, d.h. relationale, objektorientierte oder objektrelationale Schemata zugrunde liegen.

Dieser Ansatz ist recht aufwendig, insbesondere sobald neue Quellen hinzu kommen, da dann unter Umständen das globale Schema abgeändert oder sogar der gesamte Integrationsprozess erneut durchgeführt werden muss.

Solche Ansätze werden auch als *global-as-view (GAV)* bezeichnet. Das globale Schema stellt also eine Sicht auf die zugrundeliegenden Quellen dar. Diese Ansätze nehmen eine geschlossene Welt an (closed-world assumption), d.h. alle benötigten Informationen sind in

den zugrundeliegenden Quellen vorhanden und müssen nur noch in einem globalen Schema zusammengetragen werden.

In Integrationsarchitekturen ist die andere Vorgehensweise allerdings weiter verbreitet: Dabei wird zunächst ein Domänenmodell entwickelt, mit welchem der gesamte Anwendungsbereich einheitlich beschrieben wird. Das Domänenmodell kann als semantisches Netz, als Ontologie oder auch als objektrelationales Schema abgelegt werden. Dieser Ansatz ist im Wesentlichen eine Modellierungsaufgabe und kann als Top-Down-Ansatz beschrieben werden, der zunächst unabhängig von der Wahl der zugrundeliegenden Quellen durchgeführt werden kann. Hier wird also eine offene Welt angenommen (open-world assumption). Anschließend kann anhand dieses Domänenmodells der Beitrag jeder zugrundeliegenden Informationsquelle beschrieben werden, eine Quelle ist dann sozusagen eine Sicht auf das Domänenmodell. Das bezeichnet man als *local-as-view (LAV)*. Die beiden Bezeichnungen GAV und LAV wurden in [FLM98] eingeführt und werden mittlerweile häufig verwendet, um die zahlreichen existierenden Architekturen zumindest grob zu klassifizieren. Nicht immer wird ein Domänenmodell auch als solches benannt. Beispielsweise wird im Information Manifold Projekt [LRO96b] das verwendete Domänenmodell als „world view“ bezeichnet.

Eine neu hinzukommende Quelle kann bei einem LAV-Ansatz relativ einfach integriert werden. Man benötigt lediglich eine Quellenbeschreibung der neuen Quelle, diese hat keinen Einfluss auf die bereits vorhandenen Quellenbeschreibungen. Die Vorteile von LAV bei der Erweiterbarkeit werden durch eine kompliziertere Anfragebearbeitung – wenn auch mit mehr Freiheitsgraden – gegenüber GAV ausgeglichen. Eine gute Gegenüberstellung, wenn auch noch nicht mit diesen Begriffen, findet man in [Ull97]. Neuere Ansätze arbeiten auch mit hybriden Modellen, z.B. [FLM99] oder [RS01].

Die vorliegende Arbeit verfolgt den Ansatz des Domänenmodells. Der Benutzer soll ein für ihn angemessenes und auch stabiles Modell der Information rund um den Literaturbereich präsentiert bekommen, zumal mit häufigen Quellenzu- und -abgängen zu rechnen ist und wir demnach auch von einer offenen Welt ausgehen. Im folgenden Abschnitt werden nun die Anforderungen an ein Domänenmodell für den Bereich der Literaturrecherche formuliert.

5.3 Anforderungen an das Domänenmodell

Für die vorliegende Arbeit wollen wir ein Domänenmodell verwenden. Da die Erstellung im Wesentlichen eine Modellierungsaufgabe darstellt, formulieren wir zunächst die Anforderungen an ein solches Domänenmodell.

5.3.1 Anforderungskatalog

Angemessener Abstraktionsgrad: Im Bereich der Literaturrecherche sind viele Informationen zu den Dokumenten in den Katalogen verzeichnet. Oft sind diese Angaben

sehr fein granular aufgeschlüsselt, z.B. wird bei der Titelangabe in vielen Katalogen zwischen Hauptsachtitel, Titelzusatz, Zitiertitel oder Sachtitel in Ansetzungsform unterschieden. Das Domänenmodell soll ein Dokument mit den Eigenschaften beschreiben, die für einen Benutzer von Interesse sind und deren Bedeutung er nachvollziehen und beurteilen bzw. auch unterscheiden kann. Dazu sollen die Merkmale in atomaren Attributen abgelegt werden, an die Bedingungen gestellt werden können oder die zur Sortierung oder Bewertung herangezogen werden können.

Abdeckungsbreite: Bei diesem Punkt geht es um die Art und den Umfang der Attribute, die zum Beschreiben eines Dokuments angegeben werden sollen. Zur Unterstützung dieses und des vorangegangenen Punktes sollen Benutzerstudien herangezogen werden, um Aufschlüsse über die von den Benutzern genannten und berücksichtigten Merkmale zu erhalten. Im Wesentlichen soll durch die Attribute der Bereich der *klassischen bibliographischen Metadaten* abgedeckt werden. Dann sollen zusätzliche Attribute angeführt werden, die über den *Inhalt* des Buches genauere Auskünfte geben können. Verlage bzw. Buchhändler stellen häufig Zusammenfassungen, Rezensionen, das Inhaltsverzeichnis oder Kapitelauszüge zu einem Dokument zur Verfügung. Schließlich sind auch *Beschaffungsinformationen* wichtig, d.h. zu welchem Preis und zu welchen Bedingungen der Benutzer das nachgewiesene Dokument dann auch beziehen kann. Bei der Modellierung soll dabei nicht nur auf die Anforderungen des Benutzers geachtet werden, sondern auch darauf, welche Merkmale von den zugrundeliegenden Diensten geliefert werden können, um allzu viele Nullwerte zu vermeiden.

Repräsentation von verschiedenen Publikationstypen: Bei mehreren zugrundeliegenden Diensten sind in einer Ergebnismenge auch verschiedene Publikationstypen zu erwarten, z.B. gleichermaßen Monographien und Zeitschriftenartikel. Diese lassen sich zum Teil durch dieselben Merkmale, zum Teil aber auch durch spezielle individuelle Merkmale beschreiben. Wenn möglich, soll das Domänenmodell für alle Dokumentarten anwendbar sein, d.h. sozusagen einen Kern von allgemeingültigen Attributen umfassen und noch einige Erweiterungen, die nur bei Bedarf geliefert werden.

Repräsentation als DTD: Das Domänenmodell soll in Form einer DTD formalisiert werden, so dass die Ergebnismenge und die Ergebnisdokumente als XML-Dokumente gemäß dieser DTD angegeben werden können. Für die Modellierung der Ergebnismenge einer Literaturrecherche sind diese Darstellungsmittel in jedem Fall ausreichend. Auf semantisch reichhaltigere Modelle wie semantische Netze oder objektorientierte Ansätze kann verzichtet werden.

Die ersten beiden Anforderungen bedürfen noch einer weiteren Konkretisierung. Es gibt Studien darüber, welche Eigenschaften bzw. Merkmale von Dokumenten für einen Benutzer entscheidungsrelevant sind. Im Folgenden (Abschnitt 5.3.2) soll anhand einer Studie noch einmal zusammengetragen werden, welche Angaben für Benutzer wichtig und

entscheidungsrelevant sind. Aus den Äußerungen der Benutzer kann zum einen der vorhandene Abstraktionsgrad beurteilt werden, zum andern kann auch die Abdeckung der genannten Attribute gefordert werden, sofern sie in existierenden Katalogen vorhanden sind. Daher werden im nächsten Schritt (Abschnitt 5.3.3) die existierenden Dienste in ihren Anfrage- und Ergebnismodellen überprüft. Dabei können wir herausfinden, welche der zuvor identifizierten Attribute tatsächlich beschafft werden können und ob eventuell noch zusätzliche verfügbare Attribute vom Domänenmodell abgedeckt werden sollten, die ebenfalls für die Entscheidungsfindung des Benutzers relevant sein können.

5.3.2 Relevanzstudie

Die Idee, Kriterien zu untersuchen, die zwar nicht immer auch als Anfrage formuliert werden können, aber dennoch die Entscheidung über die Relevanz eines Dokuments bestimmen, ist nicht neu. Dazu gab es in den letzten Jahren zahlreiche Studien, die die Relevanzentscheidungen von Benutzern genauer unter die Lupe genommen haben. Einen Überblick über die Benutzerstudien mit den daraus gezogenen Schlussfolgerungen liefert [Wan97], welche sozusagen als Meta-Studie bezeichnet werden kann.

Die Benutzer wurden in ihrer natürlichen Umgebung beobachtet, es waren meist Studenten oder Akademiker bei der Anfertigung einer wissenschaftlichen Arbeit, z.B. Seminar-, Diplom- oder Doktorarbeit. Dabei hat sich gezeigt, dass Benutzer viele Kriterien in ihre Relevanzbeurteilung und Entscheidung mit einbeziehen, die nicht thematischer Natur sind, z.B. die Autorität des Autors, den Zitierungsstatus oder das Kosten-Nutzen-Verhältnis einer Beschaffung. Meist werden bei einer Entscheidung nur wenige und zudem immer andere (Kombinationen von) Kriterien herangezogen, je nach Kontext. Der Benutzer wird dabei durch situative, kognitive und psychologische Eigenschaften beeinflusst. Die gefundenen Studienergebnisse stützen sich gegenseitig, auch wenn die Konzepte nicht immer identisch benannt wurden. Es wurden auch Kriterien identifiziert, die nicht direkt in elektronischen Bibliothekskatalogen abrufbar sind, sondern vom Benutzer selbst ab- bzw. hergeleitet wurden. Bei den Studien wurde die Relevanz daran festgemacht, ob ein Dokument beschafft wurde bzw. ob ein Dokument schließlich in der eigenen angefertigten Arbeit zitiert wurde.

Insgesamt konnte eine kritische Menge von 17 Kriterien identifiziert werden, die immer wieder genannt wurden. Diese werden nun dargestellt, und zwar unterteilt in fünf verschiedene Schwerpunkte, wobei die Unterteilung von der Autorin dieser Arbeit selbst stammt.

Einen Schwerpunkt bilden thematische Gesichtspunkte. Diese lassen sich recht allgemein und objektiv anhand der Dokumente selbst feststellen. Ein anderer Schwerpunkt ist in erster Linie vom betreffenden Fachgebiet abhängig, d.h. in jedem Fachgebiet sind bestimmte Arten von Dokumenten wichtiger und bedeutsamer als andere. Wieder einen anderen Schwerpunkt machen die individuellen Eigenschaften des Recherchierenden aus, wobei wir hier

unterscheiden zwischen Eigenschaften, die an der Person festgemacht werden können und über eine längere Zeit und über verschiedene Recherchen hinweg konstant bleiben, und situativen Gegebenheiten, die von einer Recherche zur nächsten auch variieren können. Schließlich betrachten wir den aktuellen individuellen Problemkontext, den Anlass und das Ziel der Recherche, was ebenso die Beurteilung durch den Benutzer beeinflussen kann.

Unter den thematischen Gesichtspunkt lassen sich folgende Merkmale einordnen:

- *Thema bzw. Titel (topicality):* Dieses Attribut beschreibt das Thema des Dokuments und ist das am häufigsten herangezogene Kriterium zur Beurteilung der Relevanz.
- *Fachbereich (discipline):* Der Fachbereich ist deutlich weiter abgesteckt als das Thema, er wird aber häufig herangezogen, wenn das Thema von verschiedenen Disziplinen untersucht wird oder interdisziplinär ausgerichtet ist.
- *Klassiker (classic/founder):* Als Klassiker wird ein Werk bezeichnet, das in einem Bereich als erstes einen bedeutenden Beitrag (Theorie, Modell, Methode) erzielt hat und somit als grundlegend und wegweisend für weitere Forschungsarbeiten betrachtet werden kann.
- *Standardwerk (well-known/standard reference):* Als Standardliteratur bezeichnet man ein Dokument, häufig ein Lehrbuch, in welchem die allgemein bekannten Konzepte aus einem Bereich zusammengefasst werden.
- *Zielgruppe (orientation/level):* Dieses Attribut charakterisiert entweder den Inhalt des Dokuments präziser, z.B. als anwendungsbezogen, theoretisch oder einführend, oder beschreibt den Leserkreis, an den es sich richtet, z.B. Studenten, Wissenschaftler oder Anwender.

Vom speziellen Fachbereich hängen folgende Attribute ab:

- *Aktualität (recency):* In manchen Fachbereichen ist ein 3 Jahre altes Dokument unter Umständen schon nicht mehr aktuell, z.B. im Bereich der Informatik, in anderen Fachbereichen, wie z.B. den Geschichtswissenschaften, haben gerade alte Bücher eine große Bedeutung.
- *Autorität (authority):* Der Autor eines Dokuments kann viel zur Beurteilung der Relevanz beitragen, vor allem wenn es ein bekannter oder berühmter Autor ist. Darüber hinaus kann man den Status einer Autorität auch daran festmachen, wie viel dieser Autor sonst noch auf diesem Gebiet publiziert hat.
- *Qualität (expected quality):* Es gibt einige Anhaltspunkte, anhand derer man auf die Qualität eines Dokuments schließen kann, z.B. wenn eine Veröffentlichung einen Review-Prozess durchlaufen hat oder in einer renommierten Zeitschrift erschienen ist.

Folgende Eigenschaften sind individuell geprägt und dauern über einen längeren Zeitraum an:

- *Vorkenntnisse (cognitive requisite):* Die Beurteilung eines Dokuments erfolgt häufig aufgrund des individuellen Vor- oder Hintergrundwissens in diesem Bereich, d.h. in Abhängigkeit davon, ob ausreichend Vorkenntnisse vorhanden sind.
- *Beziehung zum Autor (relationship/origin):* Die Beurteilung ist oft davon abhängig, in welcher Beziehung der Benutzer zu dem Autor steht, z.B. ob der Autor sein Professor, sein Kollege, etc. ist.
- *Besondere Voraussetzungen (special requisite):* Hier wird beurteilt, ob spezielle Kenntnisse (z.B. Sprachkenntnisse) oder technische Voraussetzungen (z.B. Mikrofilmlesegerät, Faxanschluss, schneller Internetzugang) nötig sind, um das Dokument lesen zu können.

Folgende Attribute hängen von der aktuellen und individuellen Situation der Recherche bzw. des Benutzers ab:

- *Verfügbarkeit (availability):* Die Verfügbarkeit ist zwar für alle Benutzer eines Dienstes oder einer Bibliothek gleich, allerdings kommt hier beispielweise noch die individuelle Wertung dazu, z.B. ob 2 Tage Wartezeit lang oder kurz sind oder ob die Beschaffung in einem bestimmten elektronischen Format entweder möglich oder angenehm oder eher umständlich durchführbar ist.
- *Zeitrahmen (time/effort)*: Dieses Kriterium spiegelt sozusagen das Kosten-Nutzen-Verhältnis zum Beschaffen des Dokuments aus Sicht des Benutzers wider. Ein Benutzer wird ein bestimmtes Dokument nur weiterverfolgen, wenn dieses Verhältnis für ihn positiv ausfällt.
- *Neuheit (novelty):* Dieses Kriterium gibt an, ob das Dokument oder auch der Inhalt bereits bekannt ist für den Benutzer. Die Bewertung kann allerdings unterschiedlich ausfallen, manchmal sind nur noch neue Informationen interessant, manchmal wird ein Benutzer auch gerne an etwas Bekanntes, in Vergessenheit Geratenes erinnert oder er sucht von vornherein etwas Bestimmtes, das er schon kennt.

Die folgenden Kriterien sind vom individuellen Problemkontext geprägt. Diese Kriterien wurden häufig im Zusammenhang damit genannt, ob ein Dokument zitiert wurde oder nicht. Dem Zitieren geht natürlich eine Beschaffung und ein Durchlesen des Dokuments voraus.

- *Vorgaben durch die Form (norm):* Hierbei bestimmt die Art des zu verfassenden Dokuments über den Zitierungsumfang, der die Schwelle festlegt, ab wann ein vorgefundenes Dokument zitiert wird. Ein gewisser Rahmen und Umfang ist üblich, dieser will aber meistens auch nicht vom Benutzer überschritten werden.

- *Beurteilungsinstanz (judge):* Hierbei richtet sich die Entscheidung, ob ein Dokument zitiert wird, danach, von wem die eigene Arbeit beurteilt werden wird und wie diese Beurteilungsinstanz über das fragliche Dokument vermutlich urteilen wird.
- *Empfehlung (credential):* Hierbei werden eher „politische" bzw. „strategische" Entscheidungen getroffen, z.B. untermauert es die eigene Kompetenz, wenn gewisse Werke auch zitiert werden und damit als zur Kenntnis genommen und berücksichtigt gewertet werden. Manchmal werden auch Projektpartner aus Prinzip zitiert oder konkrete Empfehlungen oder Hinweise vom Betreuer in jedem Fall aufgenommen.

In Abbildung 5.1 sind die genannten Merkmale und Kriterien noch einmal schwerpunktartig aufgeführt.

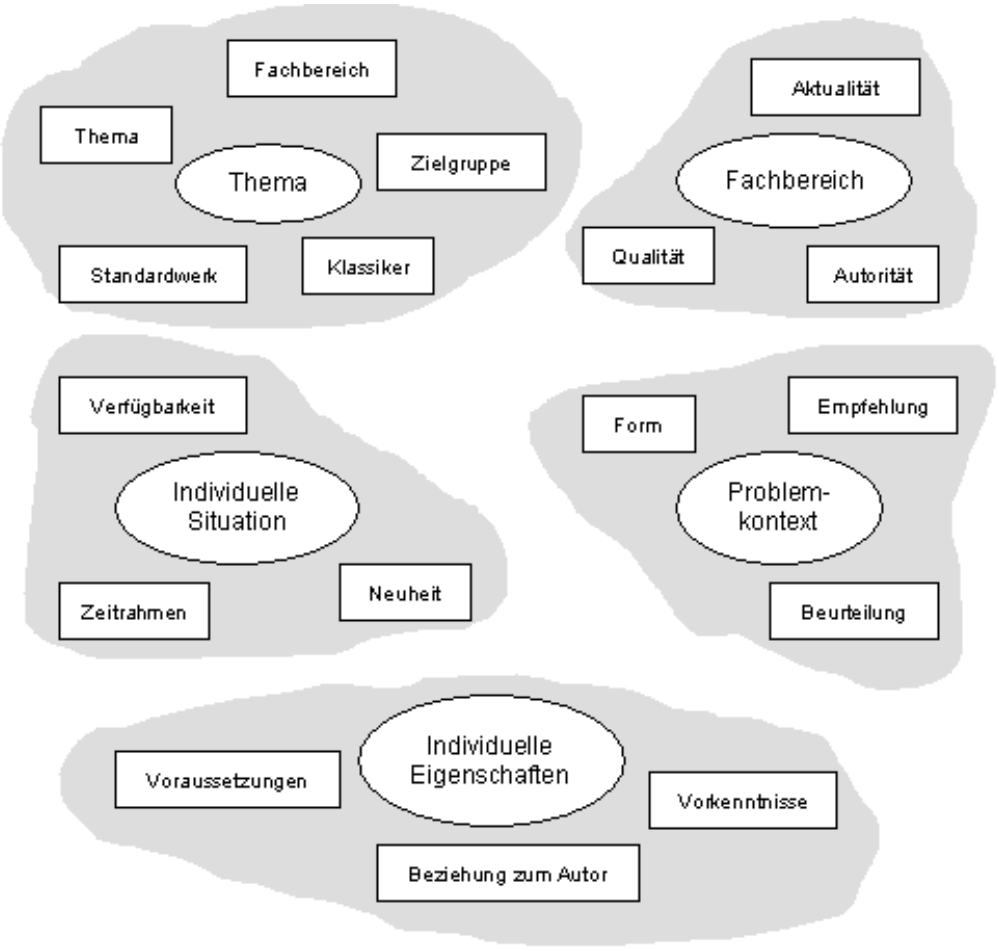


Abbildung 5.1: Übersicht über die Beurteilungskriterien der Benutzer

Zum Zeitpunkt der Studien waren nicht alle Attribute in existierenden Recherchesystemen auch explizit verfügbar. Einige Attribute davon werden heute mittlerweile in Katalogen angeboten (z.B. Verfügbarkeit, Zielgruppe). Darüber hinaus bieten heutige Systeme auch

noch weitere Merkmale oder Informationen an, die – wären sie zum Zeitpunkt der Studie bereits verfügbar gewesen – sicherlich auch genannt worden wären. Diese Merkmale wollen wir in unserem Domänenmodell natürlich auch berücksichtigen. Daher untersuchen wir im nächsten Abschnitt die aktuellen Dienste auch speziell im Hinblick darauf.

5.3.3 Informationsumfang existierender Recherchedienste

In diesem Abschnitt wollen wir anhand existierender Dienste (s. Abschnitt 2.2.1) untersuchen, welche Attribute für das Domänenmodell prinzipiell zur Verfügung stehen. Die Dienste im Literaturbereich sind nur zu einem gewissen Maße heterogen, da sie sozusagen ein One-World-Szenario bilden. Jeder Dienst beschreibt eine Menge von Dokumenten, hält aber für die Beschreibung einen unterschiedlichen Satz von Attributen und Informationen vor.

Die existierenden Dienste kann man im Großen und Ganzen in drei Dienstklassen einteilen [SO02a]: Bibliotheken, Buchhändler und Artikeldienste (mit/ohne Volltext). Die Dienstklassen unterscheiden sich zum einen in der Art der vorgehaltenen Dokumente und zum anderen in der Menge der verfügbaren Informationen zu einem Dokument.

Der Grundkonsens der existierenden Dienste besteht in einer Reihe von *bibliographischen Attributen*. Dazu gehören Informationen über Titel, Autor, Jahr, Verlag, ISBN, Auflage, Seitenzahl, Sprache und zusätzliche Erscheinungsvermerke. Dazu zählen ebenfalls die vergebenen Stich- und Schlagworte.

Jeder Dienst liefert auch – mehr oder weniger explizit – *Beschaffungsinformationen*. Bei Bibliotheken kann die Ausleihbarkeit eines Buches festgestellt werden, d.h. entweder der nächstmögliche Zeitpunkt für eine Ausleihe oder die Bestimmung, dass das Dokument nur im Lesesaal eingesehen werden kann. Buchhändler geben den Preis des Buches und mögliche zusätzliche Lieferkosten sowie den Zeitrahmen an, mit dem ein Kunde für die Lieferung rechnen muss. Neuerdings ist es auch möglich (AMAZON), gefundene Dokumente gebraucht und damit billiger zu beziehen. Dienste, die Volltexte zur Verfügung stellen, bieten oft unterschiedliche Formate an (ps, pdf), meist aber einheitliche Regelungen für die Nutzung. Beispielsweise sind Technische Berichte in der Regel kostenlos zu beziehen und die Benutzer des Karlsruher Universitätscampus haben freien Zugang zu allen Artikeln der ACM-DL (Campuslizenz). Die Lieferung eines Artikels über Subito kostet für Studenten beispielsweise für 4 Euro per Mail, 6 Euro per Post und 7 Euro per Fax.

Buchhändler oder Verlage liefern meist noch diverse *Zusatzinformationen* zu den Dokumenten, beispielsweise das Titelbild, eine Zusammenfassung oder Beschreibung des Inhalts, den Klappentext, manchmal das Inhaltsverzeichnis oder Textauszüge, Informationen zum Autor, den Verkaufsrang des Dokuments, eventuell den Platz in der aktuellen Bestsellerliste, Bewertungen und Kritiken von Lesern. Weitere Zusatzinformationen liefern Dienste wie ResearchIndex oder ACM-DL, die die Referenzen zwischen einzelnen Dokumenten nutzbar machen.

5.3.4 Konkretisierung der Anforderungen

Die klassischen bibliographischen Informationen sollten auf jeden Fall Eingang in das Domänenmodell finden. Sie decken sich größtenteils auch gut mit den Kriterien, die von den Benutzern explizit genannt wurden, wie z.B. Titel (Titel/Thema), Autor (Autorität), Jahr (Aktualität), Auflage (Aktualität), Verlag (Qualität), Publikationstyp/Ausgabe (Qualität, Form), Sprache (Voraussetzungen, Vorkenntnisse, Form) sowie Stich- und Schlagworte (Thema, Fachbereich). Auch die nicht genannten bibliographischen Daten sollten im Domänenmodell berücksichtigt werden. Die ISBN ist wichtig, da über diese Nummer ein Dokument eindeutig identifiziert werden kann, und die Seitenzahl lässt – zumindest in extremen Fällen – Vermutungen über den abgedeckten Inhalt und den Detaillierungsgrad zu.

Die Beschaffungsinformationen sind für Benutzer ebenso wesentliche Entscheidungskriterien (Verfügbarkeit), daher sollten sie mit in das Domänenmodell aufgenommen werden. Dabei sollten besonders auch Alternativen deutlich werden, d.h. nicht nur in Bezug auf Preis und Lieferzeit, sondern auch was die Formate der Lieferung (Form, Voraussetzungen) und den Ort, an den geliefert werden kann, betrifft.

Zum Zeitpunkt der Benutzerstudien waren viele der inhaltlichen Zusatzinformationen noch nicht verfügbar. Der Erfolg der Buchhändler und auch der Referenzen-Dienste untermauert auch noch einmal den Wert, den diese Angaben mittlerweile für den Benutzer haben. Sie sollten also auch im Domänenmodell nicht fehlen. Zusätzliche Beschreibungstexte oder das Inhaltsverzeichnis können weitere Aufschlüsse über das Thema und die Zielgruppe des Dokuments liefern, Rezensionen oder Leserbewertungen geben Hinweise auf die Qualität. Der Verkaufsrang kann dabei helfen, Standardwerke zu identifizieren. Zum Auffinden von Klassikern oder Autoritäten können die Referenziert-Beziehungen nützlich sein. Das Titelbild eines Dokuments sollte auch berücksichtigt werden, möglicherweise in Form einer URL, da der Benutzer daran möglicherweise erkennen kann, ob er bereits Assoziationen zu dem Dokument hat oder es vielleicht sogar bereits kennt (Neuheit).

Die bisherigen Betrachtungen konnten den Abdeckungsbereich des Domänenmodells konkretisieren. In Bezug auf den Abstraktionsgrad wählen die Benutzer in der Regel gröbere Begriffe als in den Katalogen vorgefunden werden. So kann man sicherlich „Verfügbarkeit“ noch deutlich feiner aufschlüsseln in Lieferzeit, -kosten, -adresse und -format. Das „Thema“ kann man sicherlich auch an Titel, Untertitel bzw. an den Stich- oder Schlagworten feststellen. Dass den Benutzern der Unterschied zwischen Stich- und Schlagworten geläufig ist, ist eher nicht anzunehmen. Auch das Konzept des Verlags setzt sich genau genommen aus dem Verlagsnamen und dem Verlagsort zusammen. Hier wollen wir als Anhaltspunkt für einen angemessenen Abstraktionsgrad diejenigen Attribute explizit (atomar) modellieren, anhand derer der Benutzer das Ergebnis ggf. auch sortieren würde. D.h. sowohl die Verfügbarkeit als auch der Verlag sollten im Domänenmodell weiter aufgeschlüsselt werden.

5.4 Bekannte Literatur-Beschreibungsmodelle

In diesem Abschnitt stellen wir einige bekannte und weit verbreitete Beschreibungsmodelle für Literaturdatensätze vor, um zu überprüfen, ob sie sich als Domänenmodell für die vorliegende Arbeit eignen würden. Diese Beschreibungsmodelle bestehen aus einer Menge von Attributen zur Charakterisierung von Dokumenten, sozusagen um die Metadaten zu einem Dokument zu erstellen. Hierbei ist der Beschreibungsformalismus, die Syntax sowie die technische Umsetzung für die Überprüfung der Eignung nicht von Interesse. Wichtig sind für diese Überlegungen lediglich die Modellierungsaspekte, d.h. welche Attribute verwendet werden und ob wir möglicherweise ein Modell (mit kleineren Erweiterungen oder auch Auszüge des Modells) für unser Domänenmodell übernehmen können. In einem darauf folgenden Schritt würden wir das gefundene Modell als DTD formalisieren und verwenden.

Beginnen wollen wir mit den Formaten aus dem klassischen Bibliotheksumfeld, MAB, MARC und Z39.50. Dazu liefert [Eve99] eine sehr ausführliche Beschreibung und einige Hintergrundinformationen. Dann betrachten wir eine Entwicklung aus dem digitalen Bibliotheksumfeld (Dublin Core, [DC99]) und schließlich eine weit verbreitete Methode in der wissenschaftlichen Literaturpraxis (Bibtex [Pat88], als Teil von Latex [Lam95]).

MAB: MAB ist das *M*aschinelle *A*ustauschformat für *B*ibliotheken. Mit MAB können alle im Bibliotheksbereich erzeugten Daten ausgetauscht werden: bibliographische Daten, Norm- und Lokaldaten. MAB2 (1995 verabschiedet) enthält alle erforderlichen inhaltlichen, strukturellen und technischen Erweiterungen, um MAB auch als Austauschformat in Online-Umgebungen einsetzen zu können. Die MAB-Formate definieren eine Menge von Attributen zum Beschreiben von Dokumenten. Alleine sind diese Attributdefinitionen noch nicht ausreichend, wichtig ist auch der Inhalt, der in diese Attribute dann eingetragen werden muss. Dafür gibt es die RAK (*R*egeln für die *A*lphabetische *K*atalogisierung), z.B. für öffentliche Bibliotheken (RAK-ÖB) oder für wissenschaftliche Bibliotheken (RAK-WB). Viele Deutsche Bibliotheken halten sich an MAB und RAK, da sie dadurch die Daten besser gemeinsam nutzen und austauschen können, beispielsweise in den Bibliotheksverbünden.

MARC: Der Name USMARC wurde 1999 abgeschafft, da die kanadischen (CANMARC), australischen (Australian MARC) und schließlich auch britischen Versionen (UKMARC) mit der US-Version vereinheitlicht wurden. Jetzt heißt es nur noch MARC oder MARC 21 (*MA*chine-*R*eadable *C*ataloguing). Auch hier gibt es zu RAK das angloamerikanische Pendant AACR2 (*A*nglo-*A*merican *C*ataloguing *R*ules), ein Regelwerk, in dem genau festgelegt wird, wie die Attribute eines Dokuments aufgebaut sein und aussehen sollen.

UNIMARC: UNIMARC (*UNI*versal *MARC*) ist ein internationales bibliographisches Format. Dieses Format folgt dem Ziel, den Datenaustausch zwischen nationalen bibliographischen Einrichtungen zu erleichtern. Die zugrundeliegende Idee besteht darin, dass jede Nationalbibliothek nur die Konvertierung vom eigenen Format in UNIMARC und zurück programmieren muss und dadurch eine Menge Zeit und Arbeit einsparen kann. Allerdings ist

es so, dass die gelieferten Daten je nach Herkunft qualitativ sehr unterschiedlich sind und in gewissem Umfang immer noch eine individuelle Aufbereitung notwendig ist.

In der folgenden Tabelle 5.1 ist ein Vergleich der beschriebenen Formate zu sehen.

Name	Anzahl Felder	Dokumentationsumfang	Hauptverbreitung
MAB1/MAB2	~ 700	> 700 S.	Deutschland
UNIMARC	~ 200	> 500 S.	Europa
MARC	~ 330	> 1000 S.	USA, UK, Kanada, Australien

Tabelle 5.1: Vergleich bekannter Beschreibungsmodelle für Literatur in Bibliotheken

Insgesamt ist es schwierig, Aussagen über den Umfang oder den Grad der Detaillierung der Formate zu machen. Der Umfang der Dokumentation gibt vielleicht eher einen Eindruck, allerdings sagt die Seitenzahl hier auch nur begrenzt etwas aus, da sie von der Bedruckung und vom Anteil der Wiederholungen abhängt. Die Anzahl der Felddefinitionen alleine besagt nicht allzu viel, da zum Teil viele Kategorien Mehrfachfelder sind und andererseits die Vielfalt der Unterfelder auch häufig in der Praxis gar nicht ausgeschöpft wird.

Etwas aufschlussreicher ist eine Studie der Library of Congress (LoC) darüber, welche dieser vielen Felder denn nun wirklich ausgefüllt werden. Das Ergebnis ist in Tabelle 5.2 zu sehen, sie zeigt die am häufigsten ausgefüllten Felder mit der zugehörigen Prozentangabe. Die nicht aufgeführten Felder werden in weniger als 1 % der Dokumente benutzt.

100 %	Titel, Erscheinungsvermerk (Ort, Verlag, Jahr), physikalische Beschreibung (Seitenzahl, etc.), Klassifikation (im LoC Format), Sprache
95 %	Sachschlagwort
72 %	Erster Verfasser
67 %	ISBN, Fußnote (allg. Anmerkung)
63 %	DCC (Dewey Decimal Classification)
50 %	Regionalschlüssel
49 %	Bibliographische Beschreibung
43 %	Beteiligte Personen
25 %	Geographisches Schlagwort (Coverage)
< 20 %	Ausgabevermerk, beteiligte Körperschaft, Reihe/Serie, Personenname als Schlagwort, weitere Titel (Nebentitel, etc.), Einheitstitel, Primärkörperschaft, Körperschaftsname als Schlagwort
< 5 % und >= 1 %	Abstract, Einheitstitel, Band- und Kapitelangaben, Veranstaltung, freies Schlagwort, Genre als Schlagwort, Titel als Schlagwort

Tabelle 5.2: Gebrauch der Beschreibungsfelder in der Praxis

Z39.50: Z39.50 ist ein internationales Protokoll zur Vernetzung von Bibliotheks- und Datenbankanwendungen. Es unterstützt im Allgemeinen Information-Retrieval-Anwendungen, speziell aber auch die Recherche in einem Bibliothekskatalog. Es ist wohl das bekannteste einheitliche Rechercheprotokoll (verbindungs- und sitzungsorientiert), das momentan im Bibliotheksumfeld existiert. Es ist ein ANSI/ISO-Standard (aktuell: Version 3 seit 1995), allerdings wird dieser Standard in der Praxis (noch) nicht von jeder Bibliothek unterstützt, da das Protokoll sehr komplex und aufwendig zu realisieren ist. Für den normalen Benutzer ist es darüber hinaus nahezu nicht zu bedienen und muss mindestens mit einer graphischen Eingabeunterstützung versehen werden. Daher betreiben die meisten Bibliotheken ihr individuelles (graphisches) Recherchesystem. Gut geeignet ist das Protokoll zum Integrieren von verschiedenen Katalogen, die dieses Protokoll unterstützen. Für unsere Zwecke können wir es als Anhaltspunkt heranziehen, und zwar in Kombination mit dem zugehörigen Attribute Set Bib-1. Diese Attributmenge besteht aus sechs Bereichen von Attributen, wobei nur der erste Bereich (Use Attributes) für die Beschreibung von Dokumenten verwendet wird. Es besteht aus 100 im Standard festgelegten Attributen und 175 (nachträglich) genehmigten erweiterten Attributen zum Anfragen von DC (s.u.), GILS (Global Information Locator Services) und Musikdaten. Zur Titelaufnahme stehen in den Standardattributen z.B. 15 verschiedene Attribute zur Auswahl.

Dublin Core Element Set: Im Bereich der Digitalen Bibliotheken findet dieses Modell recht häufig Verwendung. Vollständig wird es bezeichnet als Dublin Core Metadata Element Set, kurz sagt man auch Dublin Core (DC). Wie der Name bereits ausdrückt, handelt es sich dabei um eine minimale Menge von Attributen (an der Zahl 15, s.u.), die Publikationen und andere Ressourcen standardisiert beschreiben sollen. Der Fokus liegt bei diesem Modell auf der Beschreibung von Online-Dokumenten und Online-Ressourcen. Dieses Attributmodell wurde beim OCLC/NCSA Metadata Workshop 1995 erstmalig verabschiedet und wird seither noch weiter verfeinert. Die aktuelle Version ist 1.1 aus dem Jahre 1999. DC ist ein Minimalkonsens, wobei ein Minimum kein Optimum sein muss.

Folgende 15 Attribute sind zur Beschreibung eines Dokuments vorgesehen: Titel, Verfasser (Hauptautor), Schlagworte, Beschreibung (z.B. Abstract), Verleger, beteiligte Personen (außer dem Hauptautor), Datum (meist wird die Erstellung des Datensatzes genommen), Typ (Art des Dokuments), Format (z.B. MIME-Format), Identifikator (z.B. URI, URL, URN, DOI, ISBN, ISSN, ...), Quelle (die das Dokument vorhält), Sprache, Beziehung (zur Quelle oder zu anderen Dokumenten), Abdeckung (engl. coverage, zeitlich, geographisch, thematisch, ...) und Rechte (beschreiben die Rechtslage, Copyright, Rechte der Quelle, aber auch die für einen Benutzer relevanten Lese- oder Druckrechte können hier spezifiziert werden).

Bibtex: Eigentlich kommt Bibtex aus der wissenschaftlichen Welt und wird bei Latex-Dokumenten zur Erstellung des Literaturverzeichnisses und der Verweise verwendet. Man kann zwischen verschiedenen Stilen wählen, im Sinne einer Trennung von Semantik und Layout. Bibtex ist auf die klassischen Dokumente im wissenschaftlichen Bereich ausgerichtet.

Es gibt 14 verschiedene Eintragsarten, d.h. vorgesehene Publikationstypen wie Artikel, Buch, Konferenzband, Diplom-, Doktorarbeit oder Technischer Bericht. Zu jeder Eintragsart gibt es einen Satz von geforderten, optionalen und unberücksichtigten Attributen. Die häufigsten Attribute sind Autor, Titel, Editor, Band, Nummer, Seiten, Jahr, Monat, Adresse und Verlag. Manche Attribute schließen sich auch gegenseitig aus, wie z.B. Autor und Editor. Weiterhin gibt es die generische Klasse Misc und das frei verfügbare Attribut Note.

Aufgrund der Trennung von Semantik und Layout liegt der Vergleich mit XML nahe. Tatsächlich gibt es mittlerweile auch mehrere Abbildungen von Bibtex auf XML, z.B. BibteXML (http://carol.wins.uva.nl/~zegerh/bibteXML/, http://xml.coverpages.org/bibteXML .html). BibteXML ist die bekannteste Abbildung und mit den meisten anderen Ansätzen kompatibel.

Bewertung: Wir wollen nun überprüfen, ob die vorgestellten bibliothekarischen Datenformate den zuvor formulierten Anforderungen entsprechen. Dabei sind die ersten beiden Anforderungen, nämlich ein angemessener Abstraktionsgrad und eine Abdeckungsbreite, von Bedeutung.

Bezüglich eines angemessenen Abstraktionsgrades fallen die klassischen bibliothekarischen Datenformate (MAB, MARC, UNIMARC und Z39.50) aufgrund ihrer großen Anzahl (175-700) an Attributen heraus. Das verdeutlicht noch einmal, dass diese Formate für Experten, d.h. Bibliothekare, entwickelt worden sind. Außerdem werden sie in erster Linie für interne Dokumentationszwecke verwendet und nicht für die Präsentation von Dokumenten an der Benutzungsschnittstelle. Die Studie der tatsächlich benutzten Aufnahmefelder ist hingegen deutlich relevanter. Die genannten Attribute stimmen auch im Wesentlichen mit den von uns zuvor formulierten Merkmalen der bibliographischen Datensätze überein. DC hat mit 15 Attributen einen zu geringen Abstraktionsgrad, besonders für die klassischen bibliographischen Daten ist er nicht detailliert genug. Bibtex hat einen angemessenen Abstraktionsgrad.

Die Abdeckung der Attribute ist bei den vorgestellten Datenformaten nicht in allen drei von uns geforderten Bereichen vollständig gewährleistet. Die klassischen bibliographischen Attribute werden mit Ausnahme von DC von allen vorgestellten Datenformaten gut abgedeckt. Im Bereich der Zusatzinformationen bieten die Datenformate meist nur das Attribut Zusammenfassung an. Wünschenswert wären hier sicherlich noch Felder für Rezensionen, Inhaltsverzeichnisse, etc. DC bietet immerhin die Modellierung von Beziehungen zwischen Dokumenten an, mit der man gut das Zitierverhalten ausdrücken könnte. Beschaffungsinformationen sind in den klassischen Bibliotheksformaten nicht zu finden. Bibtex führt ein Attribut Adresse, mit welchem immerhin die URL eines online verfügbaren Dokuments angegeben werden kann, und DC erlaubt die Spezifikation einer Quelle und von Benutzungsrechten.

Obwohl also viele Datenformate im klassischen und im digitalen Bibliotheksbereich existieren, ist keines davon geeignet, einen angemessenen Satz von Attributen für das gesuchte Domänenmodell zu liefern. Auch die umfangreichen Datenformate können nicht geeignet reduziert werden, da sie im Bereich der Beschaffungsinformationen und der zusätzlichen Inhaltsinformationen stark ergänzt werden müssten. Daher wird im folgenden Abschnitt ein eigenes Domänenmodell entwickelt und vorgestellt, welches speziell auf die formulierten und konkretisierten Anforderungen zugeschnitten ist.

5.5 Entwurf des Domänenmodells als DTD

In diesem Abschnitt stellen wir den Entwurf, genauer gesagt das Ergebnis des Entwurfs eines eigenen Domänenmodells vor. Die Attribute, die in das Domänenmodell aufgenommen wurden, leiten sich aus den Ergebnissen der Relevanzstudie und der Beobachtung existierender Dienste ab (vgl. Abschnitt 5.3.4). Das Domänenmodell wird in Form einer DTD festgelegt. Dieselbe DTD soll für verschiedene Arten von Dokumenten (Monographien, Zeitschriften, Artikel, Technische Berichte, etc.) benutzt werden können.

Modellierungsgrundsätze: Im Wesentlichen gibt es zwei Konstrukte in XML, Elemente und Attribute. Attribute wurden immer dann verwendet, wenn ein einzelnes Merkmal in der Beschreibung eines Dokuments auftauchen kann oder nicht. Bei komplexeren Merkmalsstrukturen sind Elemente notwendig, die sich aus einzelnen Merkmalen zusammensetzen können.

Insgesamt wurde die DTD für die Ergebnismenge so modelliert, dass sie sowohl für die Gesamtergebnismenge passend ist als auch für jedes gelieferte Teilergebnis. Dabei ist es sogar möglich, die Gesamtergebnismenge nach einer Duplikaterkennung und -verschmelzung mit diesem Schema zu repräsentieren. Dazu müssen die Duplikate allerdings durch ISBN-Gleichheit gebildet werden. Um beliebige Duplikatbildungen zu unterstützen, müssten lediglich bei den einfachen Attributen im bibliographischen Teil Mehrfachnennungen erlaubt werden. Im Rahmen dieses Abschnitts wollen wir den Hauptteil der DTD vorstellen, der zur Repräsentation eines Dokuments verwendet wird. Die gesamte DTD beschreibt zusätzlich noch das Konstrukt der Ergebnismenge und enthält einige intern verwendete Steuerinformationen, die zum Verschmelzen der Teilergebnisse benötigt werden. Die vollständige DTD ist in Anhang C abgedruckt.

```
<!ELEMENT Dokument ( BibliogrInfo? , ZusatzInfo? , BeschaffungsInfo* )>
```

In Anlehnung an die in den Anforderungen identifizierten drei Bereiche von Beschreibungsmerkmalen unterteilt sich auch das Domänenmodell in drei Bereiche: die bibliographischen Informationen, zusätzliche inhaltliche Informationen und Beschaffungsinformationen. Insgesamt charakterisieren wir ein Dokument mit 28 Eigenschaften bzw. Konzepten. Die gewählten Merkmale wollen wir nun im Folgenden jeweils im Rahmen des zugehörigen Bereichs vorstellen und näher erläutern.

```
<!ELEMENT BibliogrInfo ( Titel? , Autoren? , Schlagworte? )>
  <!ELEMENT Titel ( Originaltitel? , Hauptsachtitel? , Titelzusatz? )>
    <!ELEMENT Originaltitel (#PCDATA)>
    <!ELEMENT Hauptsachtitel (#PCDATA)>
    <!ELEMENT Titelzusatz (#PCDATA)>
  <!ELEMENT Autoren ( Autor+ | Herausgeber+ | Koerperschaft )>
    <!ELEMENT Autor ( Vorname? , Nachname )>
    <!ELEMENT Herausgeber ( Vorname? , Nachname )>
      <!ELEMENT Vorname (#PCDATA)>
      <!ELEMENT Nachname (#PCDATA)>
    <!ELEMENT Koerperschaft (#PCDATA)>
  <!ELEMENT Schlagworte ( Schlagwort* , Systematik* )>
    <!ELEMENT Schlagwort (#PCDATA)>
    <!ELEMENT Systematik ( Systematikname , Kategorie+ )>
      <!ELEMENT Systematikname (#PCDATA)>
      <!ELEMENT Kategorie (#PCDATA)>
  <!ATTLIST BibliogrInfo Jahr CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Verlagsname CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Verlagsort CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Auflagenummer CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Auflageart CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Ausgabe CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Pubtyp (Buch|Artikel|...|DA) #IMPLIED>
  <!ATTLIST BibliogrInfo UebergeordnetesWerk CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Sprache CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Seiten CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Begleitmaterial CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo ISXN CDATA #IMPLIED>
  <!ATTLIST BibliogrInfo Preis CDATA #IMPLIED>
```

Zuerst gehen wir auf den Bereich der *bibliographischen Informationen* ein. Er setzt sich aus 16 Merkmalen zusammen, wobei die Konzepte Titel, Autoren und Schlagworte noch feiner aufgeschlüsselt werden. Sämtliche Merkmale sind als optional gekennzeichnet, da sie aufgrund der vorhandenen Informationsportionierung der Dienste kontinuierlich in der Ergebnismenge eintreffen.

Titel: Bei Titeln ist es häufig der Fall, dass sie in Haupt- und Untertitel unterteilt werden, bibliothekarisch korrekt ausgedrückt hieße das Hauptsachtitel und Titelzusatz. Leider ist diese Unterteilung bei den Diensten nicht immer identifizierbar, so dass man oft einen Gesamttitel (Originaltitel) erhält. Daher modellieren wir alle drei Formen im Domänenmodell, wobei in erster Linie der Originaltitel ausgefüllt wird. Für die Duplikaterkennung anhand von gleichlautenden Titeln besteht dadurch zumindest die Möglichkeit, auch auf Haupttiteln zu arbeiten. Diese sollten theoretisch besser übereinstimmen, da gerade bei Titelzusätzen oft uneinheitlich verfahren wird, z.B. was Abkürzungen oder Angaben von zusätzlichen Materialien wie CDs anbelangt.

Autoren: Zu einem Dokument kann es entweder Autoren, Herausgeber oder eine Körperschaft geben. Da diese Unterscheidung auch etwas über die Art des Dokuments aussagt, modellieren wir sie explizit. Zu einem Dokument kann es mehr als einen Autor und

mehr als einen Herausgeber geben. Es gibt in der Regel nur eine Körperschaft, die entweder als Autor oder als Herausgeber zu diesem Werk auftritt. Die verschiedenen Rollen der Körperschaft sind zum einen den meisten Benutzern wohl nicht bekannt und können zum anderen aus den Katalogen auch meist nicht extrahiert werden. Daher vernachlässigen wir diese Rollenangabe in unserem Domänenmodell und führen die Körperschaft ohne einen Zusammenhang mit einer Autoren- oder Herausgeberfunktion auf.

Die Modellierung sieht den Nachnamen und optional einen Vornamen vor. Im allgemeinen führen Autorennamen in Literaturkatalogen oft zu Problemen und Missverständnissen. Zum einen können verschiedene Autoren denselben Namen haben, zum anderen gibt es für denselben Namen oft verschiedene Schreibweisen. Beispielsweise können Vornamen ausgeschrieben oder abgekürzt werden, manchmal ist der zweite Vorname zusätzlich angegeben. Gerade wenn der Nachname Sonderzeichen enthält, können diese in den Katalogen unterschiedlich aufgelöst werden. Daher ist eine Duplikaterkennung aufgrund des Autorennamens sehr fehleranfällig und sollte zumindest zusammen mit dem Titel der Dokumente durchgeführt werden. In beiden Fällen benötigt man tolerante Stringvergleiche (s. Kapitel 10).

Hier gibt es möglicherweise bald eine Abhilfe: In Anlehnung an die ISBN wurde und wird an mehreren Instituten bereits an einer International Standard Authority Number gearbeitet, z.B. existieren Ansätze wie ISAN [Del89] oder INSAN [SR00]. Das Format der Codierung ist dabei weniger das Problem. Vielmehr sind diese Ansätze bislang an den Kosten und dem Management einer internationalen Organisation zur Vergabe solcher Nummern gescheitert.

Schlagworte: Zu jedem Dokument können mehrere Schlagworte angegeben werden. Diese werden (meist manuell) nach einem kontrollierten Vokabular vergeben. Allerdings vergeben die existierenden Dienste ihre Schlagworte meist nicht nach einer einheitlichen Systematik. Nur in wenigen Fällen verwenden sie eine allgemein bekannte Systematik zum Verschlagworten, wie beispielsweise die Dezimalklassifikation von Dewey (DDC), die Subject Headings der Library of Congress (LCSH) oder das ACM Computing Classification System (ACM-CCS). Unter diesem Merkmal kann man sowohl mehrere Schlagworte als auch mehrere Kategorien einer bestimmten Systematik angeben.

Jahr: Das Erscheinungsjahr eines Dokuments ist im Gegensatz zu den vorangegangenen Konzepten nun als einfaches Attribut modelliert, allerdings auch wieder optional. Wir vermerken hier lediglich die Jahreszahl, ohne zusätzliche Angaben von Monat, o.ä.

Verlag: Meistens setzt sich die Verlagsangabe aus dem Verlagsnamen und dem Ort zusammen. Daher haben wir für die Modellierung zwei einzelne Attribute vorgesehen, ebenfalls jeweils wieder optional. Den Fall, dass mehrere Verlage gemeinsam ein Buch herausgeben, berücksichtigen wir hier nicht. Bei Artikeln wird hier der Verlag der Zeitschrift oder des Konferenzbandes angegeben, bei Technischen Berichten die Universität.

Auflage: Bücher erscheinen häufig in mehreren Auflagen. Buchhändler führen meist nur die aktuelle Auflage, in Bibliotheken sind oft auch ältere und eben mehrere verschiedene Auflagen zu finden. Das Interessante bei nachfolgenden Auflagen ist der Vermerk, ob der Inhalt aktualisiert oder erweitert wurde. Das sieht man an Zusätzen wie erw., überarb., aktual., etc. Wir modellieren daher sowohl die Auflagennummer als auch die Art der Auflage als zwei separate und optionale Attribute.

Ausgabe: Damit wird die Ausgabe eines Buches genauer beschrieben, z.B. illustriert, in Farbe oder kartoniert. Bei Zeitschriftenartikeln wird hier die genaue Band- und Heftzählung angegeben.

Publikationstyp: Umfassenderweise könnte man alle in Bibtex vorgesehenen Publikationstypen zulassen und diese Typen darüber hinaus mit ihren eigenen Attributen definieren, wie es BibteXML macht. Wir verfolgen mit unserem Domänenmodell einen anderen Ansatz und definieren einen einzigen Attributsatz, der sozusagen den gemeinsamen Nenner aller Publikationstypen bildet. Dabei haben wir folgende Publikationstypen vorgesehen, die im wissenschaftlichen Bereich von Bedeutung sind: Buch, Artikel (bei Konferenz oder Zeitschrift oder Technischer Bericht), Zeitschrift, Konferenzband und Diplomarbeit. Eine Dissertation ist sicherlich auch von Bedeutung, läuft allerdings hier unter Buch. Sicherlich können auch noch andere Typen wie Festschriften, Jahresberichte oder auch Hörbücher interessant sein. Dazu ist lediglich die Auswahlliste von Publikationstyp zu erweitern, die übrigen definierten Attribute sind auch für die Repräsentation von diesen neuen Typen gut zu nutzen.

Übergeordnetes Werk: Hier wird bei Artikeln der Name der Konferenz oder der Zeitschrift genannt. Diese Angabe könnte auch als Link zu dem entsprechenden Dokument realisiert werden, je nach der Leistungsfähigkeit des Systems, im einfachsten Fall steht hier lediglich der Name. Dieses Attribut kann auch für Buchreihen ausgefüllt werden, z.B. für die LNCS-Reihe vom Springer Verlag.

Sprache: Hier wird die Sprache des Dokuments angegeben. Auch wenn ein Dokument prinzipiell mehrsprachig sein kann, vermerken wir hier nur eine Sprache, und zwar gemäß der zweistelligen ISO 639 Notation (http://lcweb.loc.gov/standards/iso639-2/iso639jac.html), z.B. deutsch (de), englisch (en), italienisch (it).

Seiten: Dieses Merkmal bezeichnet bei Büchern die Seitenzahl, bei Artikeln die Seitenangabe, d.h. auf welchen Seiten (von - bis) sie zu finden sind.

Begleitmaterial: Hier wird beispielsweise angegeben, ob zu dem Buch eine CD o.ä. gehört.

ISXN: Bei diesem Eintrag wird die ISBN oder die ISSN angegeben, je nach Publikationstyp. Die Notation erfolgt ohne Bindestriche, d.h. der Eintrag besteht aus neun Ziffern und an der letzten Stelle steht entweder eine zehnte Ziffer oder ein X. Für die Codierung der ISBN vergleiche beispielsweise http://www.isbn-international.org/ oder Abschnitt 10.2.

Preis: An dieser Stelle wird der offizielle Ladenpreis eines Buches angegeben. In Deutschland gilt die Buchpreisbindung, so dass dieses Attribut eindeutig festgelegt ist. Dieser Neupreis eines Buches ist eine hilfreiche Information, gerade wenn man Dienste einbindet, die gebrauchte Bücher verkaufen, und wenn der Benutzer sehen möchte, um wie viel günstiger er das Buch nun bekommen kann. Daher benötigt das Domänenmodell auch noch eine weitere Preisangabe, nämlich die, die für die Beschaffung eines Dokuments gilt (s. Lieferpreis). Die Preisangabe erfolgt in Euro. Denkbar wäre natürlich auch eine Kombination aus Währungsangabe und Preis in Zahlenangabe.

Die folgenden sechs Merkmale gehören zum Bereich der *Zusatzinformationen*.

```
<!ELEMENT ZusatzInfo ( Beschr? , Beurteilungen? , Kaufrang? , Zitiert? )>
  <!ATTLIST ZusatzInfo Titelbild CDATA #IMPLIED>
  <!ATTLIST ZusatzInfo Inhaltsverzeichnis CDATA #IMPLIED>
  <!ELEMENT Beschr ( Beschreibung* )>
    <!ATTLIST Beschr Anzahl CDATA #IMPLIED>
    <!ELEMENT Beschreibung (#PCDATA)>
  <!ELEMENT Beurteilungen ( Beurteilung* )>
    <!ATTLIST Beurteilungen Anzahl CDATA #IMPLIED>
    <!ATTLIST Beurteilungen Durchschnittswert CDATA #IMPLIED>
    <!ELEMENT Beurteilung (#PCDATA)>
  <!ELEMENT Kaufrang ( Rang* )>
    <!ATTLIST Kaufrang Durchschnittswert CDATA #IMPLIED>
    <!ELEMENT Rang (#PCDATA)>
  <!ELEMENT Zitierung ( Von* , Nach* )>
    <!ATTLIST ZitiertVon Anzahl CDATA #IMPLIED>
    <!ATTLIST ZitiertNach Anzahl CDATA #IMPLIED>
    <!ELEMENT Von (#PCDATA)>
    <!ELEMENT Nach (#PCDATA)>
```

Titelbild: Hier beschränken wir uns auf die Angabe von maximal einem Titelbild, auch wenn es in verschiedenen Formaten oder Auflösungen von unterschiedlichen Diensten beim Duplikatverschmelzen gefunden werden könnte.

Inhaltsverzeichnis: Auch bei diesem Merkmal entscheiden wir uns für maximal eine Angabe eines Inhaltsverzeichnisses. Unterschiedliche Dienste stellen das Inhaltverzeichnis möglicherweise in verschiedenen Formaten bereit (z.B. direkt im html-Code oder als pdf-Datei), der Inhalt, d.h. die Kapitelgliederung, die Überschriften und die Seitenzahlen sind jedoch identisch. Die Modellierung separiert das Inhaltsverzeichnis auch von den folgenden Beschreibungstexten, da es doch von einer anderen Qualität ist und eine zuverlässige Aussagekraft besitzt.

Beschreibungen: Hier werden Zusammenfassungen, Inhaltsangaben, Klappentexte, Hintergrundinformationen über den Autor und sonstige Texte des Verlages oder des Buchhändlers aufgeführt, möglicherweise auch Textauszüge. Zum einen ist sicherlich die Zahl der verfügbaren Zusatztexte interessant, zum anderen sind es auch die Texte selbst.

Beurteilungen: Beurteilungen gibt es sowohl von Büchern als auch von Papieren. Bei Papieren haben sie meist einen offizielleren Charakter (Peer-Review), allerdings kommt man an sie selten heran. Bei Büchern gibt es meist viele Lesermeinungen, aber auch einige offizielle Buchbesprechungen. Hier haben wir wiederum zum einen die Zahl der verfügbaren Rezensionen vermerkt und zum anderen die Texte selbst. Da auch häufig eine Bewertung in Form von Zahlen und nicht von Texten gegeben wird, haben wir auch noch ein durchschnittliches Votum hinzugefügt.

Verkaufsrang: In dieser Angabe steckt eine relevante Information für den Benutzer bzw. sie lässt sich daraus herleiten, z.B. wie verbreitet und bekannt ein Buch ist (Klassiker/Standardwerk). Bei einigen Artikeldiensten wird mittlerweile auch vermerkt, wie oft auf die einzelnen Artikel zugegriffen wurde, was ja für Online-Dokumente eine ähnliche Aussage hat wie der Verkaufsrang bei Büchern.

Zitierungen: Gerade bei Artikeln ist es eine sehr hilfreiche Zusatzinformation, welches Papier von anderen Papieren zitiert wird und welche bzw. wie viele Referenzen ein Papier selbst enthält. Existierende Artikeldienste halten diese Informationen immer häufiger vor, sei es dass diese Angaben mit bibliothekarischer Unterstützung oder automatisch gewonnen wurden. Hierbei kann die Zahl der späteren Zitierungen ein Gütekriterium sein (Klassiker), die Zahl der gemachten Zitierungen kann ebenso einen Anhaltspunkt liefern (Standardwerke und Überblicksartikel enthalten meist viele Referenzen).

Die folgenden sechs Merkmale gehören zum Bereich der *Beschaffungsinformationen*. Zu einem Dokument kann es mehrere Beschaffungsalternativen geben, eine Beschaffungsalternative benötigt dabei immer eine Kombination von Angaben, nämlich Lieferzeit, Lieferkosten, Lieferart, Lieferadresse und Lieferdienst.

```
<!ELEMENT BeschaffungsInfo ( Volltext* )>
  <!ATTLIST BeschaffungsInfo Lieferzeit CDATA #REQUIRED>
  <!ATTLIST BeschaffungsInfo Lieferkosten CDATA #REQUIRED>
  <!ATTLIST BeschaffungsInfo Lieferart (Ausleih|Kauf|...|Online) #REQUIRED>
  <!ATTLIST BeschaffungsInfo Lieferadresse CDATA #REQUIRED>
  <!ATTLIST BeschaffungsInfo Lieferdienst CDATA #REQUIRED>
  <!ELEMENT Volltext ( Format , Text )>
    <!ELEMENT Format (#PCDATA)>
    <!ELEMENT Text (#PCDATA)>
```

Lieferzeit: Das ist sozusagen die Wartezeit, d.h. die Zeit, ab der das Dokument dem Benutzer zur Verfügung steht. Die Angabe geschieht mit einer präzisen Zeitangabe, bestehend aus Minuten, Stunden und Tagen.

Lieferkosten: Die Lieferkosten summieren sämtliche für den Benutzer anfallende Kosten bis zum Bezug des Dokuments auf. Im Standardfall ist das der Buchpreis zuzüglich der Lieferkosten. Beim Kauf von gebrauchten Büchern können die Lieferkosten natürlich auch unter dem offiziellen Buchpreis liegen. Diese Gesamtkosten sind für den Benutzer sicherlich

ein entscheidendes Kriterium, nach dem er die Ergebnismenge ggf. auch sortieren möchte. Daher wurde auf die explizite Modellierung der beiden Einzelposten Buchkosten und Lieferkosten verzichtet.

Lieferart: Ein Dokument kann auf verschiedene Arten bezogen werden. Der Benutzer kann ein Dokument beispielsweise in der Bibliothek einsehen oder ausleihen, er kann es kaufen, d.h. online bestellen und sich dann zuschicken lassen, oder er kann es möglicherweise (in verschiedenen Formaten) online beziehen. Genau diese vier Arten sind momentan im Domänenmodell auch verankert.

Lieferadresse: Das ist der Ort, an den das Dokument geliefert wird bzw. an den sich der Benutzer hinbegeben muss. Das kann zum einen die Bibliothek, genauer gesagt das Abholregal oder der Lesesaal sein oder eine Postadresse, z.B. der eigenen Wohnung oder des Büros.

Lieferdienst: Diese Angabe identifiziert den Anbieter der Dienstleistung. Für den Benutzer kann es von Bedeutung sein, welcher Dienst hinter einer bestimmten Lieferoption steht. Möglicherweise bevorzugt der Benutzer bestimmte Dienste.

Volltext: Wenn der Volltext geliefert werden kann, dann wird dazu der Dateityp und der Link zum Dokument angegeben. Oft stehen verschiedene Dateiformate zur Verfügung, z.B. bei ResearchIndex. Da in der Regel die Lieferkosten und auch die anderen Lieferangaben unabhängig vom gewählten elektronischen Format sind, können hier mehrere Volltext-Einträge zu denselben Lieferbedingungen gemacht werden.

Nachfolgend wird in Tabelle 5.3 überblicksartig dargestellt, wie das Domänenmodell für verschiedene Publikationstypen auszufüllen ist. Die Tabelle zeigt noch einmal die wesentlichen Konzepte, die in unserem Domänenmodell berücksichtigt worden sind. Da für jedes Dokument die Beschaffungsinformationen prinzipiell komplett ausgefüllt werden können, sind sie hier unter einem einzigen Eintrag zusammengefasst genauso wie einige eng zusammengehörige Merkmale (z.B. Verlagsname und Verlagsort).

Die „X“-Einträge besagen, dass dieses Konzept beim gegebenen Publikationstyp in existierenden Katalogen angegeben wird. „O“ bedeutet, dass diese Informationen bei existierenden Diensten (noch) eher selten nachgewiesen werden, es aber in einigen Fällen schon geschieht oder zumindest prinzipiell möglich wäre, diese Information zu erzeugen und zu präsentieren. „–“ besagt, dass die Information im Zusammenhang mit dem Publikationstyp nicht existiert.

Tabelle 5.3 macht durch die große Anzahl an „X“- und „O“-Einträgen deutlich, dass mit der entwickelten DTD tatsächlich die unterschiedlichsten Arten von Dokumenten repräsentiert werden können, so wie es in der zuvor formulierten Anforderung verlangt wurde.

In Anhang C ist neben der vollständigen DTD auch ein Beispieldokument nach der Duplikateliminierung angegeben, so dass die verschiedenen Beschaffungsalternativen deutlich werden.

	Buch	ZArtikel	KArtikel	TBericht	Zeitschrif-tenheft	Konferenz-band	DA
Titel	X	X	X	X	X	X	X
Autoren	X	X	X	X	X	X	X
Schlagworte	X	X	X	X	X	X	X
Jahr	X	X	X	X	X	X	X
Verlag	X	X	X	X	X	X	X
Auflage	X	–	–	–	–	–	–
Ausgabe	X	X	–	–	X	–	–
Publikationstyp	X	X	X	X	X	X	X
ÜbergWerk	O	X	X	–	X	–	–
Sprache	X	X	X	X	X	X	O
Seiten	X	X	X	X	–	O	O
Begleitmat.	X	–	–	–	–	–	O
ISXN	X	X	X	–	X	X	–
Preis	X	–	–	–	X	X	–
Titelbild	X	–	–	–	X	O	–
Inhaltsverz.	X	–	–	–	X	X	–
Beschreib.	X	X	X	X	X	X	X
Beurteilungen	X	O	O	O	–	–	O
Verkaufsrang	X	O	O	O	O	O	O
Zitierungen	O	X	X	X	–	–	O
Beschaffung	X	X	X	X	X	X	X
X: Information verfügbar; O: Information manchmal verfügbar; –: Information existiert nicht							

Tabelle 5.3: Merkmale von Publikationstypen

5.6 Resümee

In diesem Kapitel wurde ein eigenes Domänenmodell entworfen, nachdem dargelegt wurde, dass vorhandene Beschreibungsmodelle aus dem Literaturbereich die Forderungen nach einem angemessenen Abstraktionsgrad und einer breiten Abdeckung der Merkmale nicht erfüllen. Das entworfene Domänenmodell orientiert sich sowohl an den verfügbaren Daten vorhandener Dienste als auch an den Ergebnissen von Benutzerstudien. Das entstandene Domänenmodell beinhaltet etwa 30 Attribute, welche die bibliographischen Angaben abdecken, zusätzliche inhaltliche und statistische Informationen liefern und konkrete Beschaffungsalternativen aufzeigen.

Das Domänenmodell wird zum einen für die einheitliche interne Repräsentation der Datensätze verwendet, zum anderen dient es dem Benutzer, um seine Anfragen zu formulieren. Der Benutzer kann Bedingungen oder Vorlieben in Bezug auf die vorgegebenen Attribute angeben und festlegen, welche der Attribute im Ergebnis angezeigt werden sollen und anhand welcher Attributen das Ergebnis sortiert werden soll. Die nächsten beiden Kapitel beschäftigen sich mit der Bereitstellung der benötigten Anfragefunktionalität. In Kapitel 6 wird eine ausdrucksmächtige Anfragesprache entwickelt, in Kapitel 7 wird eine geeignete Umsetzung der Anfragefunktionalität an der Benutzungsschnittstelle konzipiert und realisiert.

6 Spezifikation einer Anfragesprache

6.1 Überblick

Nachdem im vorangegangenen Kapitel die geeigneten und verfügbaren Informationen für ein Meta-Recherchesystem im Literaturbereich geklärt und durch die Spezifikation eines Domänenmodells formalisiert und vereinheitlicht wurden, soll im Folgenden nun eine geeignete Anfragefunktionalität zum Umgang mit diesen Informationen entwickelt werden.

Dazu werden zunächst noch einmal die zu Beginn der Arbeit formulierten Anforderungen benannt, die sich auf die Anfragefunktionalität beziehen, sowie einige ergänzende Forderungen aufgeführt, die sich bei der Betrachtung verwandter Arbeiten im Rahmen von Kapitel 3 ergeben haben (Abschnitt 6.2). Genau genommen betreffen die Anforderungen zwei unterschiedliche Aspekte der Anfragefunktionalität: Der eine Aspekt ist die prinzipiell unterstützte Anfragesprache, also die Mächtigkeit der zur Verfügung stehenden Anfrageoptionen, und der andere Aspekt betrifft die geeignete Umsetzung der Anfragefunktionalität an der Benutzungsschnittstelle. Kapitel 6 befasst sich mit der Entwicklung der vom System unterstützten Anfragefunktionalität und der Spezifikation einer geeigneten Anfragesprache. In Kapitel 7 wird anschließend für die definierte Anfragefunktionalität eine geeignete Umsetzung an der Benutzungsschnittstelle entwickelt.

Um eine geeignete Anfragesprache zu spezifizieren, wird folgendermaßen vorgegangen. Zunächst werden zwei bekannte Anfragesprachen, SQL und XQuery, anhand der gegebenen Anforderungen auf ihre Eignung überprüft (Abschnitt 6.3). Danach erfolgt die Spezifikation einer geeigneten Anfragesprache (Abschnitt 6.4). Diese soll in einem nächsten Schritt um weiche Bedingungen bzw. Präferenzen erweitert werden. Dazu werden zunächst zwei Spracherweiterungen betrachtet, die bereits die Angabe von Präferenzen erlauben, Preference SQL und Preference XPath (Abschnitt 6.5). Schließlich wird die vorgeschlagene Anfragesprache um Präferenzen erweitert (Abschnitt 6.6).

6.2 Anforderungen und Lösungsansatz

6.2.1 Anforderungen an die Anfragefunktionalität

In Abschnitt 3.1 wurden Anforderungen an ein Meta-Recherchesystem im Bereich der wissenschaftlichen Literaturrecherche und -beschaffung formuliert. Folgende Anforderungen betreffen die Anfragefunktionalität eines Meta-Recherchesystems:

- A2: Einfache, intuitive Benutzerinteraktion

- A3: Ausdrucksstarke Benutzerinteraktion und Kontrollmöglichkeit
- A4: Berücksichtigung menschlicher Fähigkeiten, Fertigkeiten und Grenzen:
  - Bandbreite
  - Individualität
  - Erinnerungshilfen
  - niedrige Erwartungen
  - Ungeduld
- A6: Duplikatbehandlung

Diese Anforderungen machen das Problem deutlich, das bei der Entwicklung einer geeigneten Anfragefunktionalität besteht: Auf der einen Seite wird für ein Meta-Recherchesystem eine ausdrucksstarke Benutzerinteraktion gefordert, d.h. eine hinreichend mächtige Anfragesprache sowie eine angemessene Bearbeitungsmöglichkeit für die Ergebnisse. Dadurch soll dem Benutzer die Möglichkeit gegeben werden, sich vor einer Informationsüberflutung zu schützen. Die Forderung nach Anfragemächtigkeit wird auch durch Teilaspekte von A4 untermauert: Eine Bandbreite von verschiedenen Anfragen soll möglich sein und die Anfragen sollen möglichst individuell und speziell gestellt werden können. Weiterhin soll der Benutzer eine Kontroll- und Steuermöglichkeit für den Vorgang der Meta-Recherche erhalten. Beispielsweise sollen dem Benutzer Erinnerungshilfen angeboten werden oder er soll die Behandlung von Duplikaten fordern und ggf. auch beeinflussen können (A6).

Auf der anderen Seite wird für das Meta-Recherchesystem eine einfache, intuitive Benutzerinteraktion gefordert (A2), die menschlichen Eigenschaften Rechnung trägt (A4). Dazu sollen keine zu hohen Erwartungen an den Benutzer gestellt werden, so dass auch neue oder ungeübte Benutzer das Meta-Recherchesystem möglichst intuitiv benutzen können.

Genau genommen betreffen die genannten Anforderungen zwei unterschiedliche Aspekte des Meta-Recherchesystems: Zum einen die prinzipiell unterstützte Anfragesprache, also die Mächtigkeit der zur Verfügung stehenden Anfrageoptionen, und zum anderen die geeignete Umsetzung der Anfragefunktionalität an der Benutzungsschnittstelle.

Der vorgeschlagene Lösungsansatz orientiert sich an dieser Zweiteilung und betrachtet die beiden Aspekte im Folgenden separat: Zunächst wird eine Anfragesprache entwickelt, die die genannten Anforderungen hinsichtlich der Anfragemächtigkeit und -funktionalität erfüllt. Diese Anfragesprache kann sozusagen als textbasierte Programmierschnittstelle (API, Application Programming Interface) des Meta-Recherchesystems verstanden werden. In einem zweiten Schritt, nämlich in Kapitel 7, werden dann Konzepte entwickelt, um diese Anfragesprache mittels einer graphischen Benutzungsschnittstelle für den Benutzer geeignet umsetzen zu können. Tabelle 6.1 unterteilt die Anforderungen nochmals gemäß der Relevanz für jede Teillösung.

Anfragesprache und -funktionalität	Gestaltung der Benutzungsschnittstelle
	A2: Einfache, intuitive Benutzerinteraktion
A3: Ausdrucksstarke Benutzerinteraktion und Kontrollmöglichkeit	
A4: Menschliche Eigenschaften: • Bandbreite • Individualität	A4: Menschliche Eigenschaften: • Erinnerungshilfen • niedrige Erwartungen • Ungeduld
A6: Duplikatbehandlung	

Tabelle 6.1: Unterteilung der Anforderungen an die Benutzerinteraktion

Die bisher genannten Anforderungen können noch durch die Kriterien zur Gestaltung von textbasierten Rechercheschnittstellen (vgl. Abschnitt 3.5.2.3, Gestaltungsrahmen, [SBC98]) ergänzt werden. Der Gestaltungsrahmen fordert für eine Rechercheschnittstelle in erster Linie Klarheit und Kontrolle. Er identifiziert vier Phasen bei einer Recherche und formuliert für jede Phase spezielle Anforderungen.

Bei der *Formulierung der Anfrage* soll der Benutzer die gewünschten Quellen explizit angeben können, so dass im Falle eines Meta-Recherchesystems nicht immer alle eingebundenen Dienste bzw. eine nicht offensichtliche Auswahl an Diensten befragt werden. An dieser Stelle wäre für die Auswahl der Quellen durchaus mehr Flexibilität wünschenswert. Das Meta-Recherchesystem kann beispielsweise auf die Hilfe eines Traders zurückgreifen (Abschnitt 2.4). Daher können auch Anfragen wie „die schnellsten 3 Dienste“ oder „alle Bibliotheken vor Ort“ unterstützt werden. Zu derartigen Anfragen kann der Trader nämlich die geeigneten Dienste benennen. Die explizite Angabe der zu befragenden Dienste ist also eine Anforderung an die Anfragesprache, wohingegen die flexible Quellenauswahl an der Benutzungsschnittstelle umgesetzt und unter Zuhilfenahme des Traders realisiert werden kann.

Eine Suchanfrage soll sich auf ausgewählte Eigenschaften bzw. Merkmale von Dokumenten beziehen können (dazu stehen im vorliegenden Ansatz die Attribute des Domänenmodells zur Verfügung), soll mehrere Bedingungen mit booleschen Verknüpfungen kombinieren können und die Suchvarianten des eingegebenen Textes für den Benutzer nachvollziehbar machen. Diese Forderungen beziehen sich sowohl auf die Anfragesprache als auch auf die Gestaltung der Benutzungsschnittstelle. Der Umgang mit Suchvarianten gestaltet sich bei Meta-Recherchesystemen allerdings schwierig. Die eingebundenen Dienste bieten höchst unterschiedliche Suchvarianten an, machen diese nicht immer sichtbar und auch nicht explizit ansprechbar. Daher können sich Meta-Recherchesysteme entweder sozusagen auf den „kleinsten gemeinsamen Nenner“ zurückziehen, was im vorliegenden Fall bedeuten würde, gar keine Suchvarianten zuzulassen, oder aber selbst Funktionen zur Wortstammbildung, Stoppwortreduktion, etc. realisieren. In der vorliegenden Arbeit wird – auch im Hinblick auf

Klarheit und Nachvollziehbarkeit für den Benutzer – auf die Unterstützung von Suchvarianten verzichtet.

Die *Suchaktion* soll explizit über einen Knopf ausgelöst werden, vor allem sollen keine unterschiedlichen Auslösemechanismen oder mehrere Knöpfe für die gleiche Aktion angeboten werden. Diese Forderung bezieht sich auf die Gestaltung der Benutzungsschnittstelle.

Für die *Ergebnispräsentation* soll der Benutzer die Anzahl der gewünschten bzw. angezeigten Treffer, die dargestellten Informationen, die Reihenfolge und Gruppierung der Treffer und das Layout festlegen können. Bis auf das Layout beziehen sich die Forderungen sowohl auf die Anfragefunktionalität als auch auf die Gestaltung der Benutzungsschnittstelle. Zunächst soll in einer Anfrage eine Ausgangskonfiguration für das Ergebnis angegeben werden können (Anfragesprache), anschließend sollen die Operatoren zur Nachbearbeitung interaktiv auf der Ergebnismenge angewendet werden können (Benutzungsschnittstelle).

Für den letzten Schritt, die *Anfrageverfeinerung*, soll der Benutzer die Anfragen und Ergebnisse speichern sowie durch Bewertungen von Einzeltreffern die Anfrage verfeinern können. Die Speicherung einer Historie ist keine Anforderung an die Anfragefunktionalität. Die Anforderung betrifft eher die Gestaltung der Benutzungsschnittstelle, sie wird aber im Wesentlichen von einer separaten Komponente für die persistente Datenspeicherung erfüllt werden müssen. Auf den Einsatz von Benutzerbewertungen, die eine Anfrage automatisch verfeinern können, wird im vorliegenden Ansatz zugunsten der Klarheit und Nachvollziehbarkeit einer Anfrage verzichtet.

Tabelle 6.2 teilt die Forderungen des Gestaltungsrahmens, die für den vorliegenden Lösungsansatz berücksichtigt werden, gemäß der Relevanz für jede der beiden Teillösungen auf.

Anfragesprache und -funktionalität		Gestaltung der Benutzungsschnittstelle	
Anfragen:	Quellenauswahl (explizit), Suchfelder, Verknüpfung	Anfragen:	Quellenauswahl (flexibel), Suchfelder, Verknüpfung
		Suchaktion:	Ein Auslöser
Ergebnis:	Anzahl gewünschter Treffer Auswahl Attribute Reihenfolge Treffer Gruppierung Treffer	Ergebnis:	Anzahl angezeigter Treffer Auswahl Attribute Reihenfolge Treffer Gruppierung Treffer Layout
		Abschluss:	Speicherung der Historie

Tabelle 6.2: Ergänzung der Anforderungen aus dem Gestaltungsrahmen

6.2.2 Vorgehensweise und Lösungsansatz

Im vorangegangenen Abschnitt wurden die Anforderungen, die die Anfragefunktionalität eines Meta-Recherchesystems betreffen, nochmals zusammengestellt, ergänzt und schließlich in zwei Bereiche aufgeteilt, nämlich in Anforderungen, die die Anfragesprache betreffen, und in Anforderungen, die die Gestaltung der Benutzungsschnittstelle betreffen. Im weiteren Verlauf dieses Kapitels soll nun eine hinreichend mächtige Anfragesprache entwickelt werden, die anschließend als API für das Meta-Recherchesystem verwendet werden kann, um eine geeignete graphische Benutzungsschnittstelle für das Meta-Recherchesystem zu entwickeln. Diese API bietet zudem die Möglichkeit, mehrere verschiedene Benutzungsschnittstellen für das Meta-Recherchesystem zu realisieren.

Zunächst soll überprüft werden, ob bereits eine geeignete Anfragesprache existiert, so dass keine neue Anfragesprache definiert werden muss. Die Anfragen werden an das in Kapitel 5 entworfene Domänenmodell gestellt. Dieses weist von der Konzeption her große Ähnlichkeiten mit dem relationalen Modell auf, ist aber formal gesehen eine DTD, die das XML-Format der Ergebnisse festlegt. Im Bereich der relationalen Datenbanksysteme gibt es eine sehr bekannte, weit verbreitete und standardisierte Anfragesprache: SQL (Structured Query Language [Dat94]). Zum Anfragen von XML-Datenbeständen etabliert sich allmählich auch eine Anfragesprache: XQuery [XQU02]. Die Anfragesprache XQuery ist recht jung und befindet sich noch in einer Arbeitsversion, wird allerdings vom W3C als zukünftige XML-Anfragesprache zur Standardisierung vorbereitet.

Im ersten Schritt des Lösungsansatzes soll genauer überprüft werden, ob und inwieweit sich SQL bzw. XQuery als Anfragesprachen für das angestrebte Meta-Recherchesystem eignen und die genannten Anforderungen erfüllen (Abschnitt 6.3). Daraus ergibt sich die Spezifikation einer geeigneten Anfragesprache (Abschnitt 6.4). Falls SQL bzw. XQuery geeignet sind oder nur kleinere Änderungen vorzunehmen wären, wird sich der vorgeschlagene Lösungsansatz vorzugsweise an einer standardisierten, bekannten und bewährten Sprache orientieren und keiner eigenen neuen Anfragesprache bedürfen.

In diesem ersten Schritt wird sozusagen die Basis für eine Anfragesprache geschaffen, die die genannten Anforderungen erfüllt. Eine Grundidee des vorgeschlagenen Lösungsansatzes besteht darin (vgl. Kapitel 4), nicht nur die üblichen „harten“ Bedingungen bei der Anfrageformulierung zuzulassen, sondern zusätzlich auch „weiche“ Bedingungen, sogenannte Präferenzen, zu erlauben. Benutzerstudien haben gezeigt, dass es für Benutzer intuitiv ist, Vorlieben anzugeben, beispielsweise in Form von „lieber ... als ...“- oder „möglichst ...“-Aussagen. Benutzer geben nur ungern viele harte Bedingungen an, weil sie befürchten, dass dadurch relevante Dokumente ausgeschlossen werden.

Im Hinblick auf weiche Bedingungen gibt es bereits Spracherweiterungen, die das Formulieren von Präferenzen erlauben. Preference SQL ([KK01], [Kie02]) ist beispielsweise eine aktuelle Spracherweiterung von SQL, die es erlaubt, mit Präferenzen umzugehen.

Preference XPath [KHFH01] ermöglicht die Angabe von Pfadausdrücken, die Präferenzen enthalten. Daher soll für die Erweiterung der Anfragesprache in einem zweiten Schritt geprüft werden, ob und inwieweit sich die Sprachkonstrukte von Preference SQL und Preference XPath für die Formulierung von Vorlieben im Bereich der Literaturrecherche eignen (Abschnitt 6.5). Falls die existierenden Funktionalitäten geeignet und ausreichend sind, soll darauf zurückgegriffen bzw. darauf aufgebaut werden. Andernfalls soll die spezifizierte Anfragesprache um geeignete Funktionen und Konstrukte erweitert werden. In Abschnitt 6.6 erfolgt die Spezifikation der Spracherweiterung um Präferenzen.

6.3 Bekannte Anfragesprachen

In diesem Abschnitt werden zwei bekannte Anfragesprachen kurz vorgestellt und bewertet, SQL und XQuery. SQL ist eine Anfragesprache für relationale Datenbanksysteme, XQuery ist eine Anfragesprache für XML-Datenbestände.

6.3.1 SQL

SQL ist die bekannteste Anfragesprache im Bereich der relationalen Datenbanksysteme. Zudem ist sie standardisiert und weit verbreitet [Dat94]. SQL ist allerdings weit mehr als eine Anfragesprache. Mit SQL können sowohl Datenbasisschemata als auch Datensätze erzeugt, geändert und gelöscht werden. Weiterhin enthält SQL Mechanismen zur Vergabe von Rechten und zur Steuerung von Transaktionen. Der Teil von SQL, der sich mit dem Formulieren von Anfragen beschäftigt, ist die SELECT-Anweisung. Die SELECT-Anweisung erlaubt es, Anfragen an eine Menge von Relationen (Tabellen) zu stellen.

Bei einem Meta-Recherchesystem liegt kein eigener Datenbestand vor, da die Daten erst durch geschickte Anfragen an die eingebundenen Dienste beschafft werden müssen. In der vorliegenden Arbeit besteht der zugrundeliegende Datenbestand also nicht aus Relationen. Der Datenbestand ist nur virtuell vorhanden – etwa vergleichbar mit Sichten im relationalen Modell.

Der Datenbestand, d.h. die insgesamt verfügbare Menge an Dokumenten, kann im vorliegenden Fall durch eine einzige Sicht repräsentiert werden. Dabei wird jedes Dokument mit den in der DTD spezifizierten Merkmalen beschrieben (vgl. Abschnitt 5.5 oder C.1), die Merkmale entsprechen folglich den Spalten der Sicht. Bei einem Szenario, in dem der gesamte Datenbestand nur in einer einzigen Relation vorliegt, bezeichnet man diese Relation als *universelle Relation*. Auch wenn es sich in unserem Fall genau genommen um eine *universelle Sicht* handelt, werden wir im Folgenden von der universellen Relation sprechen, da der Anfragemechanismus im relationalen Modell für Relationen und Sichten vollkommen identisch ist.

Das Besondere der vorliegenden universellen Sicht besteht darin, dass sie auch mehrwertige und zusammengesetzte Attribute beinhaltet. Diese entstehen bei der Abbildung einer XML-Datenstruktur auf ein relationales Modell. Beispielsweise können zu einem Dokument mehrere Autoren oder mehrere Beschaffungsalternativen angegeben werden (vgl. C.2). Der SQL2-Standard (SQL-92) sieht nur atomare Attribute vor. Im Jahre 1999 wurde der SQL3-Standard verabschiedet (heute: SQL:99), welcher auch mehrwertige und zusammengesetzte Attribute berücksichtigt. SQL:99 geht sogar noch weiter und sieht die Definition von Objekten und Methoden vor. Diese Funktionalität wird zur Darstellung der universellen Relation und zur Formulierung von Anfragen nicht benötigt.

In diesem Abschnitt soll nun überprüft werden, ob sich SQL als Anfragesprache für den vorliegenden Meta-Rechercheansatz eignet. Die Funktionalität von SQL, die die Formulierung von Anfragen betrifft, also die SELECT-Anweisung, soll hier nur in groben Zügen beschrieben werden. Als standardisierte Datenbankanfragesprache wird SQL als bekannt vorausgesetzt. Für detailliertere Informationen verweisen wir auf Lehrbücher ([Dat94], [Ebn02], [Kli01], [For99]) oder den Standard selbst ([SQL92], [SQL99]). Betrachtet wird hier die Anfragefunktionalität des SQL-92-Standards, erweitert um die notwendigen Konstrukte zum Umgang mit mehrwertigen und zusammengesetzten Attributen. Nach der kurzen Übersicht wird die SELECT-Anweisung anhand der in Abschnitt 6.2 genannten Anforderungen bewertet.

Eine SELECT-Anweisung besteht aus verschiedenen Teilen, die durch spezielle Schlüsselworte eingeleitet werden. Eine Anfrage ist vollständig, wenn mindestens der SELECT- und der FROM-Teil angegeben werden. Alle weiteren Teile sind optional.

```
  SELECT    [DISTINCT] <projektionsausdruck>
  FROM      <tabellenspezifikation>
[ WHERE     <selektionsbedingung>             ]
[ GROUP BY  <spaltenspezifikation>
[ HAVING    <gruppenbedingung>                ]]
[ ORDER BY  <sortierung>                      ];
```

SELECT: Der Projektionsausdruck reduziert eine Relation auf die ausgewählten Attribute (Spalten). Die Angabe von * führt sämtliche Spalten der Relation auf. Auf die Attribute können arithmetische Operationen (+, -, *, /) oder Aggregatfunktionen (z.B. Anzahl, Summe, Durchschnitt, Minimum, Maximum) angewendet werden. Wird nach SELECT das Schlüsselwort DISTINCT angegeben, werden die Duplikate im Ergebnis eliminiert. Als Duplikate werden dabei Datensätze betrachtet, die in allen ausgegebenen Attributen dieselben Werte aufweisen. Die Ausgabe von zusammengesetzten Attributen erfolgt über die Punktnotation oder mithilfe der zugehörigen Konstruktormethoden. Mehrwertige Attribute können ebenfalls über die zugehörige Konstruktormethode oder separat über Cursor ausgegeben werden.

FROM: Die Tabellenspezifikation benennt die zu verwendenden Relationen. Von den genannten Relationen wird das Kartesische Produkt gebildet. Die dadurch entstehende Relation weist sämtliche Spalten der beteiligten Tabellen auf und verknüpft jeden Datensatz einer Tabelle mit allen Datensätzen der anderen Tabellen.

WHERE: Die Selektionsbedingung richtet sich an die Datensätze (Zeilen) der Relation. Nur die Datensätze, die die Bedingung erfüllen, werden im Ergebnis ausgegeben. Die Bedingungen beziehen sich auf die Attribute der Relationen. Bedingungen können numerische und alphanumerische Vergleichsoperatoren enthalten sowie Existenz- und Universalquantoren verwenden. Mehrere Bedingungen können mit booleschen Operatoren verknüpft werden. Zur Formulierung von Bedingungen können auch Unteranfragen generiert werden. Unteranfragen bestehen wiederum aus einer SELECT-Anweisung und stellen neben dem Kartesischen Produkt eine zweite Möglichkeit dar, um Datensätze aus verschiedenen Tabellen miteinander zu verknüpfen.

GROUP BY: Die Spaltenspezifikation gibt eine Reihe von Attributen an, anhand derer das Ergebnis gruppiert werden soll. Die entstandenen Gruppen enthalten in den genannten Attributen genau einen Wert. In den anderen Attributen treten mehrere Werte auf, die allerdings aufgrund der Forderung nach atomaren Domänen nicht ausgegeben werden können. Zur Ausgabe können lediglich die genannten Attribute oder Aggregatfunktionen (s. SELECT) auf den anderen Attributen verwendet werden. Wird eine Spaltenspezifikation bei GROUP BY angegeben, beziehen sich Aggregatfunktionen immer auf die entstandenen Gruppen, ansonsten werden Aggregatfunktionen für die gesamte Relation berechnet.

HAVING: Eine Gruppenbedingung ist eine Selektionsbedingung für Gruppen. Nur die Gruppen, die die Bedingung erfüllen, werden im Ergebnis ausgegeben. Die Bedingungen können Vergleichsoperatoren auf Attributen bzw. Aggregatfunktionen von Attributen enthalten, mehrere Bedingungen können wiederum boolesch verknüpft werden. HAVING kann nur angegeben werden, sofern vorher GROUP BY benutzt wurde.

ORDER BY: Die Sortierung benennt die Attribute, nach denen das Ergebnis auf- bzw. absteigend sortiert werden soll. Dazu können nur Attribute verwendet werden, die auch in der SELECT-Klausel ausgegeben werden. Die verwendeten Attribute können auch berechnete oder aggregierte Werte enthalten.

Bewertung

A3: SQL ist eine sehr ausdrucksstarke Anfragesprache, sie erlaubt die Spezifikation beliebig komplexer Anfragebedingungen. Der Benutzer hat viele Möglichkeiten, Angaben über das gewünschte Ergebnis zu machen. Die Kontrolle über die Ausführung der Anfrage hat der Benutzer allerdings nicht. Das kommt daher, dass SQL eine deskriptive Anfragesprache ist, bei der der Benutzer nur das gewünschte Ergebnis spezifizieren kann, nicht aber die intern gewählte Vorgehensweise zur Berechnung des Ergebnisses. An einigen Stellen ist SQL auch

zu mächtig für die vorliegende Problemstellung. Da nur Anfragen an eine universelle Relation gestellt werden müssen, wird die Kombination mehrerer Relationen nicht benötigt. Die FROM-Komponente und die Unteranfragen im WHERE-Teil sind also nicht notwendig. Auch die Berechnung von Attributen oder die Anwendung von Aggregatfunktionen ist beim vorliegenden Szenario nicht so wichtig, als dass sie jeder Benutzer individuell gestalten können sollte.

A4: Da die Anfragesprache hinreichend mächtig ist, wird sowohl eine Bandbreite als auch eine individuelle Formulierung von Anfragen unterstützt. Die Deskriptivität ist für die Kontrolle des Benutzers zwar nicht von Vorteil, dafür aber für die Formulierungen von Anfragen um so geeigneter. Der Benutzer spezifiziert lediglich das gewünschte Ergebnis und nicht die dazu notwendigen Bearbeitungsschritte. Die deskriptive Anfrageformulierung ist sicherlich einem prozeduralen Ansatz vorzuziehen, sie entspricht nicht nur besser den Fertigkeiten des Benutzers, sondern eröffnet der Anfragebearbeitung und -optimierung auch eine Reihe von Freiheitsgraden.

A6: Operatoren zur Duplikaterkennung sind vorhanden. SQL basiert auf Mengen, genauer gesagt auf Mehrfachmengen von Datensätzen. Duplikate können explizit zugelassen oder unterdrückt werden. Die Definition eines Duplikats erfolgt hier anhand der absoluten Wertegleichheit. Wir haben bereits festgestellt (vgl. Kapitel 4), dass es im Bereich der Literaturrecherche nicht eindeutig ist, welche Dokumente als Duplikate betrachtet werden sollen. Die genaue Gleichheit aller Attribute wird vermutlich weder vorkommen noch gewünscht sein. Gewünscht ist dagegen eher eine Zusammenfassung von Dokumenten, die beispielsweise gleiche Titel und Autoren, aber unterschiedliche Lieferbedingungen oder Auflagen haben. Dazu könnte die Gruppierung verwendet werden. Allerdings ist die Ausgabe der Attribute, die eine Menge von Werten und damit Alternativen enthalten, im relationalen Modell nur über Aggregatfunktionen möglich. Für den vorliegenden Ansatz sollten auch diese Mengen von Werten ausgegeben werden können.

Anfragen: Die Attribute des Domänenmodells stehen für die Formulierung der Bedingungen zur Verfügung. Verschiedene Suchfelder und boolesche Verknüpfungen von Bedingungen werden von der SELECT-Anweisung unterstützt (WHERE). Die Angabe der zu befragenden Quellen kann über eine entsprechende Selektionsbedingung auf dem Attribut Nachweisquelle nachgebildet werden. Als Bedingungen werden nur harte Bedingungen zugelassen.

Ergebnisse: Die Anzahl der gewünschten Treffer kann nicht explizit angegeben werden, dafür aber die Auswahl der Attribute (SELECT), die im Ergebnis angezeigt werden sollen, sowie die Reihenfolge der Treffer (ORDER BY), die anhand einer auf- oder absteigenden Attributfolge festgelegt werden kann. Eine Gruppierung der Treffer ist möglich (GROUP BY), allerdings ist die Ausgabe der Gruppen recht unkomfortabel, da für jedes Attribut nur ein eindeutiger Wert ausgegeben werden kann. Das betrifft in gleichem Maße auch die Ausgabe von mehrwertigen Attributen. Durch SQL:99 werden zwar zusätzliche Konstrukte

und Funktionalitäten angeboten und unterstützt, die Ausgabefunktionalität hat sich allerdings nicht entsprechend weiterentwickelt und basiert – wie der SQL-92-Standard – weiterhin auf atomaren Typen.

6.3.2 XQuery

Für XML-Datenbestände gibt es bislang keine standardisierte Anfragesprache. Allerdings wurde bereits 1999 vom W3C eine Arbeitsgruppe gebildet, die sich mit dem Entwurf und der Standardisierung einer Anfragesprache für XML-Datenbestände beschäftigt. Die vorgeschlagene Anfragesprache nennt sich XQuery. Die Spezifikation entwickelt sich immer noch weiter, bisher wurden schon eine Reihe von Entwürfen veröffentlicht. Die aktuelle Version ist XQuery 1.0, der letzte Entwurf stammt vom November 2002. XQuery verwendet die Pfadausdrücke von XPath, um die Navigation in hierarchisch strukturierten XML-Dokumenten auszudrücken. XQuery 1.0 basiert auf XPath 2.0. Im Folgenden wird nur eine knappe Übersicht über die Anfragesprache XQuery gegeben. Für detailliertere Informationen sei auf [KST02] oder [XQU02] verwiesen.

Eine Anfrage in XQuery ist eine sogenannte FLWR-Anweisung (sprich „flower“). Diese Anweisung besteht – wie die SELECT-Anweisung auch – aus charakteristischen Teilen, die jeweils durch entsprechende Schlüsselworte eingeleitet werden. Eine FLWR-Anfrage ist vollständig, wenn sie mindestens eine FOR- oder eine LET-Angabe enthält und die Struktur des Ergebnisses spezifiziert.

```
( FOR     <variable> IN <ausdruck>
        (,<variable> IN <ausdruck>)*  |
  LET     <variable> := <ausdruck>
        (,<variable> := <ausdruck>)*  )+
[ WHERE   <bedingung>                 ]
  RETURN  <ergebnisstruktur>
```

FOR: In diesem Teil – wie auch im folgenden LET-Teil – werden Variablen an einen Wert oder an mehrere Werte gebunden. Diese Werte können beispielsweise über Pfadausdrücke gewonnen werden. Die FOR-Klausel wird immer dann verwendet, wenn eine Iteration über eine Knotenmenge benötigt wird. Werden mehrere FOR-Klauseln, d.h. Variablenbindungen angegeben, erfolgt die Iteration über jede der Knotenmengen. Das führt zu einer Folge von Datensätzen, in welcher jede Wertekombination der Variablen auftaucht (vergleichbar mit der Bildung des Kartesischen Produkts). Bei der Variablenbindung an eine Menge von Werten können mit der Angabe von DISTINCT gleiche Werte aus der Menge eliminiert werden.

LET: Auch durch die LET-Klausel können Variablen gebunden werden. Der Unterschied zur FOR-Klausel besteht darin, dass mit der LET-Klausel jede Variable nur an einen Wert gebunden wird. Die LET-Anweisung weist einer Variable einen Knoten, eine Knotenliste oder einen berechneten Ausdruck zu. Zur Spezifikation können arithmetische Operationen und Aggregatfunktionen verwendet werden. Iterationen erfolgen nicht durch die LET-Klausel.

Aus der unterschiedlichen Syntax für die Variablenbindung (`FOR … IN …` bzw. `LET … := …`) lässt sich die unterschiedliche Semantik auch intuitiv erkennen.

WHERE: Durch die Bedingung werden genau die Datensätze ausgewählt, die das Auswahlkriterium erfüllen. Bedingungen können sich auf die eingeführten Variablen beziehen und auch Existenz- und Universalquantoren enthalten. Mehrere Bedingungen können boolesch verknüpft werden.

RETURN: Durch die Ergebnisstruktur wird bestimmt, in welcher Form die Datensätze, die die angegebenen Bedingungen erfüllen, zurückgegeben werden. Für jeden Datensatz wird ein Ergebniselement mit den entsprechenden XML-Markierungen (Elementkonstruktoren) aufgebaut. Dabei kann sich das ausgegebene Ergebnis durchaus von den Eingangsdaten unterscheiden, beispielsweise im Inhalt oder in der Struktur. Beim Aufbau der Ergebniselemente sind auch Fallunterscheidungen erlaubt. Mit einer `IF-THEN-ELSE`-Konstruktion kann die gewählte Ausgabestruktur sogar von Bedingungen abhängig gemacht werden. Zusätzlich können in die Ergebnisstruktur `SORTBY`-Klauseln eingeschoben werden. Dadurch kann ein Attribut bzw. ein Ausdruck angegeben werden, das bzw. der für jedes Ergebnis einen Wert liefert und nach welchem das Ergebnis dann auf- oder absteigend sortiert wird. Im Zuge der Standardisierungsbestrebungen wird derzeit diskutiert, ob die `SORTBY`-Konstruktion, die innerhalb der Ergebnisstruktur flexibel eingesetzt werden kann, separat ausgewiesen werden soll. Vorgeschlagen ist ein zusätzliches Schlüsselwort `ORDER BY`, welches vor der `RETURN`-Klausel eingeschoben werden soll.

Bewertung:

A3: Die FLWR-Anweisung ist sehr ausdrucksstark. In den Anfragen können beliebig komplexe Bedingungen spezifiziert und miteinander verknüpft enthalten. XQuery ist eine funktionale Anfragesprache, die es dem Benutzer erlaubt, das gewünschte Ergebnis zu spezifizieren, ihm aber keine Kontroll- bzw. Steuermöglichkeiten für die Ausführung der Anfrage an die Hand gibt. Für die Optimierungsspielräume der Anfragebearbeitung ist das wiederum von Vorteil. Die Angabe mehrerer Variablenbindungen mit der FOR-Klausel kann dazu verwendet werden, verschiedene Datensätze miteinander zu verknüpfen, indem das Kartesische Produkt gebildet wird. Für das vorliegende Szenario, das im Wesentlichen aus einer großen Menge von Datensätzen besteht, die jeweils ein Dokument beschreiben, wird diese Funktionalität nicht benötigt.

A4: Da XQuery hinreichend mächtig ist, wird sowohl eine gewisse Bandbreite als auch die individuelle Formulierung von Anfragen unterstützt. Die Formulierung der benötigten Variablenbindungen und Pfadausdrücke bedarf allerdings einiges Geschicks und einiger Übung, da die benötigten Ausdrücke rasch sehr komplex werden können. Das betrifft zwar den Benutzer nicht direkt, stellt aber deutlich höhere Anforderungen an die Transformation einer Benutzeranfrage in den entsprechenden FLWR-Ausdruck.

A6: Um Duplikate unter den Treffern zu identifizieren, stehen bei XQuery folgende Möglichkeiten zur Verfügung. In der FOR-Klausel können durch das Schlüsselwort DISTINCT doppelte Werte aus den zu durchlaufenden Mengen entfernt werden. Dadurch werden Duplikate von Beginn an unterdrückt und können im Folgenden dann auch nicht mehr herangezogen werden, um beispielsweise deren Beschaffungsinformationen mit anderen Beschaffungsoptionen zu vergleichen. Eine andere Möglichkeit der Duplikaterkennung besteht darin, die Bedingungen beim Aufbau der Ergebnisstruktur zu nutzen. So können zu einem Dokument auch die Beschaffungsalternativen der Duplikate ausgegeben werden, wobei sogar flexibel festgelegt werden kann, wie die Bedingung für Duplikate lauten soll.

Anfragen: Durch die DTD unseres Domänenmodells sind die Attribute vorgegeben, an die Bedingungen gestellt werden können. Dabei muss für mehrwertige und zusammengesetzte Attribute des Domänenmodells allerdings die Baumstruktur berücksichtigt werden. Die zu befragenden Quellen können dadurch angegeben werden, dass Bedingungen an das Attribut Nachweisquelle gestellt werden. Bedingungen können mit booleschen Operatoren verknüpft und mit Existenz- und Universalquantoren versehen werden.

Ergebnisse: Im RETURN-Teil können die Teile des Datensatzes, d.h. die Attribute, die zur Ergebnisanzeige gewünscht werden, beliebig zusammengesetzt und auch geschachtelt werden. Die Gruppierung der Ergebnisse kann mit Hilfe einer geschachtelten Ausgabe geschehen. Die Ergebnisse können nach flexibel spezifizierbaren Kriterien sortiert werden. Die Anzahl der gewünschten Ergebnisse lässt sich bei einem FLWR-Ausdruck nicht angeben, es sei denn die Ausgabe wird an einem gewissen Punkt abgebrochen.

6.3.3 Fazit

In Bezug auf die geforderte Mächtigkeit sind SQL und XQuery recht ähnlich und vollkommen ausreichend. Bei der Entwicklung von XQuery sind viele Konzepte von anderen Anfragesprachen mit eingeflossen, so auch zahlreiche Konzepte von SQL. In Hinblick auf die benötigte Funktionalität sind beide Anfragesprachen eher zu mächtig, was beispielsweise die Kombination von mehreren Relationen bzw. Datenmengen oder die Berechnung von Attributen betrifft. Hier könnte die unterstützte Funktionalität auch reduziert werden. Eine Kontroll- bzw. Steuermöglichkeit für den Benutzer bieten beide Anfragesprachen nicht. Für die Anfragebearbeitung und -optimierung ist die deskriptive bzw. funktionale Anfrageformulierung hingegen von Vorteil.

Die Bandbreite und Individualität von Anfragen wird aufgrund der Ausdrucksmächtigkeit von beiden Sprachen gleichermaßen unterstützt. Für Benutzer sind beide Sprachen bei weitem zu kompliziert, Anwendungsprogrammierer sind vermutlich mit beiden Sprachen in ähnlicher Weise vertraut. Die Spezifikation von Variablenbindungen und Pfadausdrücken ist bei XQuery deutlich aufwendiger, SQL erlaubt hingegen recht prägnante und kompakte Formulierungen.

Eine flexible Duplikaterkennung kann bei SQL durch die Gruppierung aufgrund von Wertegleichheit in bestimmten Attributen durchgesetzt werden, allerdings wird die Ausgabe von mehreren Werten in einem Attribut nicht unterstützt. XQuery kann über die flexible Strukturierung der Ergebnismenge Duplikate erkennen und entsprechend darstellen. Prinzipiell sind dadurch sogar selbst gewählte Definitionen von Duplikaten anwendbar. Allerdings gestaltet sich die notwendige Formulierung und Realisierung sehr aufwendig, da FLWR-Ausdrücke im Grunde genommen dafür nicht vorgesehen sind.

SQL- und XQuery-Anfragen können verschiedenartige Bedingungen an die Attribute des Domänenmodells stellen und dadurch auch die zu befragenden Quellen spezifizieren. Die Ausdrucksmächtigkeit beider Sprachen ist in diesem Punkt in etwa identisch.

SQL- und XQuery-Anfragen erlauben die Auswahl von Attributen, die im Ergebnis aufgeführt werden sollen. Bei mehrwertigen Attributen bereitet die Ausgabe in SQL allerdings Schwierigkeiten, hier müssen die entsprechenden Konstruktormethoden verwendet werden oder separate Ausgaben über Cursor in Kauf genommen werden. Die Ergebnisse können mit beiden Anfragesprachen nach bestimmten Attributen auf- bzw. absteigend sortiert werden. Die Sortierungsmöglichkeiten sind jedoch bei XQuery insofern mächtiger, als nach beliebigen Werten bzw. Ausdrücken sortiert werden kann, auch wenn diese im Ergebnis gar nicht erscheinen. In SQL kann nur nach Attributen sortiert werden, die im Ergebnis auch ausgegeben werden. Die Gruppierung der Ergebnisse wird von SQL direkt unterstützt, XQuery bietet hingegen eine flexible Ergebnisstrukturierung an, die für diesen Zweck – wenn auch in recht aufwendiger Weise – verwendet werden kann. Da für das vorliegende Szenario eine einheitliche Ergebnismengenstruktur verwendet werden soll, wird die bedingte Ausgabefunktionalität von XQuery dafür nicht benötigt.

Die genannten Anforderungen werden von beiden Anfragesprachen recht gut und in etwa im gleichen Maße erfüllt. Allerdings kann keine der Sprachen direkt, d.h. ohne Änderungen bzw. Erweiterungen, für die vorliegende Arbeit verwendet werden, gerade was eine flexible Duplikaterkennung oder Gruppierungsfunktionalität betrifft.

Für SQL und XQuery stehen Implementierungen zur Verfügung. Diese Implementierungen gehen jeweils von vollständig vorliegenden Datenbeständen aus. Im vorliegenden Szenario müssen die benötigten Daten allerdings erst durch Anfragen an die zugrundeliegenden Dienste portionsweise angefordert werden. Bei der Abstützung auf eine existierende Implementierung müsste also der zentrale Bereich der Anfragebearbeitung neu konzipiert und auf das vorliegende Szenario abgestimmt werden. Dadurch, dass keine existierende Implementierung ohne Veränderungen übernommen werden kann, muss aber auch nicht an sämtlichen Konzepten der vorgestellten Sprachen festgehalten werden, besonders nicht an denen, die für das vorliegende Szenario nicht benötigt werden oder ungeeignet sind.

Dennoch wollen wir uns an einer bekannten Sprache orientieren, da sich beide vorgestellten Sprachen zum einen bewährt haben und zum anderen zukünftigen API-Programmierern auch

bereits bekannt sein dürften. Die Spezifikation der benötigten Anfragesprache soll sich an SQL orientieren. SQL ist bereits standardisiert, etabliert und weit verbreitet, während bei XQuery in Zukunft noch einige Änderungen zu erwarten sind (beispielsweise im Hinblick auf die Sortieroption). SQL hat darüber hinaus den Vorteil von kompakten Formulierungen, da keine Variablen gebunden werden müssen.

Im folgenden Abschnitt soll nun die benötigte Anfragefunktionalität in Anlehnung an SQL spezifiziert werden. Dabei sollen notwendige Erweiterungen – beispielsweise im Hinblick auf die Ausgabe von mehrwertigen Attributen und die flexible Duplikaterkennung – eingebracht werden sowie auf nicht benötigte Funktionalitäten – beispielsweise zum Verknüpfen von mehreren Relationen – verzichtet werden.

6.4 Spezifikation einer geeigneten Anfragesprache

In diesem Abschnitt erfolgt nun die Spezifikation einer Anfragesprache in Anlehnung an SQL. Diese Anfragesprache soll im Folgenden als MLS-QL bezeichnet werden. Der Zusatz QL (*Q*uery *L*anguage) soll den Bezug zu SQL andeuten, MLS steht für *M*eta-*L*iteratur*S*uche bzw. *M*eta *L*iterature *S*earch. Zur Spezifikation von MLS-QL werden die vordefinierten Schlüsselworte in SQL der Reihe nach aufgegriffen. Für jedes Schlüsselwort wird untersucht, welche Funktionalität von SQL weiterhin unterstützt werden soll und welche Erweiterungen notwendig sind. Im Anschluss an diese Betrachtungen liefert Tabelle 6.3 noch einmal eine Gegenüberstellung der Funktionalitäten von SQL und MLS-QL.

SELECT: Durch die SELECT-Klausel erfolgt die Projektion des Ergebnisses auf ausgewählte Attribute. Diese Funktionalität kann für MSL-QL übernommen werden. Die Teilattribute der zusammengesetzten Attribute werden mit der Punkt-Notation angesprochen. Die *-Operation, die sämtliche Attribute anzeigt, wird beibehalten und im Zusammenhang mit der Punkt-Notation insofern erweitert, als damit sämtliche Teilattribute eines zusammengesetzten Attributs angezeigt werden können. BeschaffungsInfo.* zeigt beispielsweise die Lieferzeit, die Lieferkosten, die Lieferart, die Lieferadresse, den Lieferdienst und die vorhandenen Volltexte an. Die Beschränkung von SQL auf die direkte Ausgabe von atomaren Attributen ist für den vorliegenden Ansatz zu restriktiv. Die Ausgabe von mehrwertigen Attributen wird in MLS-QL zugelassen. Berechnete und aggregierte Attribute werden im vorliegenden Szenario nicht benötigt und daher auch nicht unterstützt. Die Funktionalität von DISTINCT ist für eine flexible Duplikaterkennung nicht ausreichend und wird in MLS-QL ebenfalls nicht angeboten. Die Duplikaterkennung wird im Sinne einer Gruppierung interpretiert und daher unter GROUP BY aufgegriffen. Die SELECT-Klausel wird in MLS-QL um den optionalen Zusatz `TOP(k)` erweitert, damit der Benutzer mit `k` angeben kann, wie viele Ergebnisse er maximal sehen möchte.

FROM: Die FROM-Klausel bzw. die Bildung des Kartesischen Produkts wird im vorliegenden Szenario nicht benötigt, da jeder Anfrage dieselbe universelle Relation zugrunde

liegt, die sämtliche verfügbaren Dokumente umfasst. Für ein intuitiveres Verständnis wollen wir an dieser Stelle die Semantik so anpassen, dass in der FROM-Klausel alle zu befragenden Quellen explizit aufgezählt werden. Die angegebenen Quellen werden dann in paralleler Weise befragt, die Ergebnisse der parallelen Anfragen werden anschließend vereinigt. Der Dienst des Traders zur automatischen Quellenauswahl kann folglich im Zusammenhang mit der Benutzungsoberfläche bzw. im Rahmen der Unterstützung der Anfrageformulierung in Anspruch genommen werden, um dadurch die Namen der zu befragenden Quellen zu gewinnen und an die Anfragebearbeitung weiterreichen zu können.

WHERE: An dieser Stelle werden die klassischen Selektionsbedingungen unterstützt, welche sich auf alle Attribute des Domänenmodells beziehen können, mit Ausnahme der Attribute, die zum Bereich der NachweisInfo gehören (ID und Nachweisquelle). Die Bedingungen bzgl. der Nachweisquellen werden durch die neue Semantik der FROM-Klausel abgedeckt, die ID eines Dokuments ist für die interne Verarbeitung eingeführt worden und für den Benutzer nicht sichtbar. Die Bedingungen können boolesch verknüpft werden. Die Formulierung von Unteranfragen, die in erster Linie zum Verknüpfen von Inhalten aus verschiedenen Relationen dient, wird für das vorliegende Szenario nicht benötigt und von MLS-QL folglich nicht unterstützt. Die Umsetzung der weichen Bedingungen wird in den folgenden beiden Abschnitten separat behandelt und im weiteren Verlauf des Kapitels ergänzt werden.

GROUP BY: Das Gruppieren kann im vorliegenden Szenario unter zwei Aspekten geschehen. Es können lose Gruppen gebildet werden, beispielsweise aus Dokumenten mit dem gleichen Erscheindungsjahr, mit einem gleichen Autor oder mit dem gleichen Verlag. Dabei soll die Ausgabe auch Attribute zulassen, nach denen nicht gruppiert wurde und die nach dem Gruppieren mehrere Werte aufweisen. Werden mehrwertige Attribute zur Gruppierung herangezogen, muss angegeben werden, ob die Gruppierung aufgrund eines gleichen Wertes durchgeführt werden soll oder nur, wenn alle Werte gleich sind. Das kann durch den Zusatz von SOME bzw. ALL vor dem entsprechenden Attributnamen gekennzeichnet werden. Die Standardbelegung ist dabei SOME. Die zweite Art der Gruppierung ist die Bildung von Duplikatgruppen. Der Unterschied zur losen Gruppierung besteht darin, dass Duplikate unter einem Stellvertreter, welcher beispielsweise dann alle verfügbaren Beschaffungsalternativen enthält, verschmolzen werden. Beide Gruppierungskonzepte können orthogonal zueinander verwendet werden. Nach der Ausführung aller Selektionsbedingungen werden auf der Menge der verbleibenden Dokumente Duplikate gebildet. Anschließend werden Gruppierungen erzeugt, in denen die Stellvertreter der Duplikate genauso wie normale Dokumente vorkommen können. Daher erweitern wir für MLS-QL die Folge der Schlüsselworte zwischen WHERE und GROUP BY um ein weiteres: DUPLICATES BY.

DUPLICATES BY: Hier werden die Attribute aufgezählt, anhand derer die Duplikate bestimmt werden sollen. Im vorliegenden Szenario reicht die absolute Wertegleichheit zur Erkennung von Duplikaten allerdings nicht aus. Gerade bei Titelaufnahmen oder Autorennamen sind Tippfehler oder unterschiedliche Schreibweisen zu berücksichtigen.

Daher ist die Unterstützung von fehlertoleranteren Zeichenkettenvergleichen wichtig. Da dies ein zentrales Thema der vorliegenden Arbeit ist, wird dieser Problematik im Folgenden ein ganzes Kapitel (Kapitel 10) gewidmet. An dieser Stelle sei vorweggenommen, dass dafür beispielsweise Trigramm-Vergleiche oder Angaben zum Editierabstand zusammen mit einem Schwellwert verwendet werden können. Diese Operatoren können mit der Angabe von USING hinter dem entsprechenden Attributnamen angegeben werden. Erfolgt keine explizite Angabe, wird der exakte Gleichheitsoperator verwendet.

HAVING: Diese Option erlaubt es, Bedingungen an Gruppen zu formulieren und damit nur bestimmte Gruppen auszuwählen. Diese Funktionalität wird im vorliegenden Szenario nicht benötigt und folglich von MLS-QL nicht unterstützt.

ORDER BY: Wie bei SQL kann das Ergebnis nach bestimmten Attributen oder Attributkombinationen auf- bzw. absteigend sortiert werden. Durch die Unterstützung von mehrwertigen Attributen, die aufgrund des zugrundeliegenden Domänenmodells, aber auch infolge einer Gruppierung auftreten können, muss die Semantik für Sortierungen von mehrwertigen Attributen definiert werden. Bei der aufsteigenden Sortierung soll der kleinste Wert einer Gruppe ausschlaggebend sein, bei der absteigenden Sortierung der größte. Die Sortierangabe kann prinzipiell mehrere einwertige und mehrwertige Attribute beinhalten. Erfolgt in der Anfrage eine Gruppierung, kann maximal ein mehrwertiges Attribut, welches aufgrund der Sortierung entstanden ist, d.h. nach welchem nicht gruppiert wurde, bei der Bildung der Sortierreihenfolge berücksichtigt werden. Nach diesem einen Attribut werden dann die Dokumente innerhalb jeder Gruppe sortiert.

Eine MLS-QL Anfrage ist dann vollständig, wenn mindestens SELECT, FROM und WHERE angegeben werden. Damit sieht der Aufbau einer MLS-QL Anfrage folgendermaßen aus:

```
  SELECT          [TOP(k)] <projektionsausdruck>
  FROM            <quellenauswahl>
  WHERE           <selektionsbedingung>
[ DUPLICATES BY   <attributname> [ USING <vergleichsoperator>
                (, <attributname> [ USING <vergleichsoperator> ] )* ]
[ GROUP BY        <attributname> [ SOME | ALL ]
                (, <attributname> [ SOME | ALL ] )*                 ]
[ ORDER BY        <sortierung>                                      ] ;
```

Folgende Tabelle stellt noch einmal überblicksartig zusammen, an welchen Stellen sich die vorgeschlagene Sprachspezifikation MLS-QL von SQL unterscheidet, d.h. an welchen Stellen sie mit weniger Funktionalität auskommt bzw. die Funktionalität erweitert wurde.

	SQL	MLS-QL
SELECT	Projektion auf einzelne Attribute	Projektion auf einzelne Attribute
	*, Auswahl aller Attribute	*, Auswahl aller Attribute, in Verbindung mit zusammengesetzten Attributen Auswahl aller Teilatttribute
	DISTINCT	–
	–	TOP(k)
	berechnete Attribute	–
	aggregierte Attribute	–
	Punkt-Notation für zusammengesetzte Attribute	Punkt-Notation für zusammengesetzte Attribute
	Darstellung von mehrwertigen Attributen mit Konstruktormethoden oder separat über Cursor	Unterstützung von mehrwertigen Attributen durch geeignete Darstellungsmittel
FROM	Bildung des Kartesisches Produkts der aufgeführten Relationen	Parallele Befragung der aufgeführten Quellen (Vereinigung)
WHERE	Selektionsbedingungen an Datensätze, die alle Attribute der Relationen betreffen können	Selektionsbedingungen an Datensätze, die alle Attribute des Domänenmodells betreffen können (mit Ausnahme von ID und Nachweisquelle)
	Boolesche Verknüpfung	Boolesche Verknüpfung
	Universal- und Existenzquantoren	–
	Bedingungen mit Unteranfragen	–
DUPLICATES BY	–	Duplikatbildung aufgrund von gleichen Attributwerten
	–	Verwendung von speziellen Vergleichsoperatoren mittels USING
GROUP BY	Gruppierung nach gleichen atomaren Attributwerten	Gruppierung nach gleichen Attributwerten (mit SOME und ALL)
	Aggregatfunktionen auf Gruppen	–
HAVING	Selektionsbedingungen an Gruppen	–
ORDER BY	auf- bzw. absteigende Sortierung anhand von ausgegebenen, berechneten oder aggregierten Attributen	auf- bzw. absteigende Sortierung anhand von ausgegebenen Attributen

Tabelle 6.3: Überblick und Vergleich des Sprachumfangs von SQL und MLS-QL

Im folgenden Beispiel sollen zwei Bibliotheken, die Universitätsbibliothek Karlsruhe (UBKA) und die Badische Landesbibliothek in Karlsruhe (BLB) sowie der Internet-

Buchhändler Amazon (AMAZON) befragt werden. Der Benutzer sucht Bücher, die im Titel Java enthalten und im Jahr 2000 oder später erschienen sind. Die Duplikate sollen anhand von gleichen Titeln und Autoren identifiziert werden. Dazu wird der Trigramm-Vergleich mit einem Schwellwert von 90 verwendet. Die Ergebnisse sollen nach dem Erscheinungsjahr gruppiert werden. Die Sortierung erfolgt hier zunächst absteigend nach den Jahreszahlen, innerhalb der Gruppe dann nach der Verfügbarkeit der einzelnen Dokumente:

```
SELECT         *
FROM           UBKA, BLB, AMAZON
WHERE          BibliogrInfo.Titel.Originaltitel LIKE "%Java%"
AND            BibliogrInfo.Jahr >= 2000
DUPLICATES BY  BibliogrInfo.Titel.Originaltitel USING TRI-90,
               BibliogrInfo.Autoren USING TRI-90
GROUP BY       BibliogrInfo.Jahr
ORDER BY       BibliogrInfo.Jahr DESC, BeschaffungsInfo.Lieferzeit;
```

Für eine leichtere Lesbarkeit können wir auch Synonyme für die Attribute des Domänenmodells einführen. Die Bezeichnung jedes Attributs ist im vorliegenden Domänenmodell eindeutig. Wir können also für jedes Attribut einen gleichnamigen Alias vergeben, der sozusagen die Navigation in der DTD verdeckt (z.B. `BibliogrInfo.Titel.Originaltitel alias Originaltitel`).

Das Attribut Autoren ist genau genommen ein mehrwertiges und zusammengesetztes Attribut. Es kann eine Körperschaft oder mehrere Autoren- bzw. Herausgebernamen beinhalten, die wiederum aus einem Vor- und einem Nachnamen bestehen (vgl. Abschnitt 5.5 oder C.1). Zur Bestimmung der Autorengleichheit wird hier das Attribut Autoren herangezogen – mit der Semantik, dass aus allen untergeordneten Einträgen eine Zeichenkette gebildet wird. Diese Zeichenkette kann dann mit dem toleranten Trigramm-Operator mit den Autorenangaben der anderen Dokumente verglichen werden.

6.5 Präferenzen in Anfragesprachen

Nachdem im vorangegangenen Abschnitt eine geeignete Anfragesprache spezifiziert wurde, soll diese nun um weiche Bedingungen bzw. Präferenzen erweitert werden. Dazu werden zunächst zwei existierende Spracherweiterungen vorgestellt, die bereits die Formulierung von Präferenzen unterstützen: Preference SQL und Preference XPath. Nach einer kurzen Übersicht über die angebotene Funktionalität soll überprüft werden, ob sich damit die Präferenzen der Benutzer im Bereich der wissenschaftlichen Literaturrecherche in geeigneter Weise ausdrücken lassen.

6.5.1 Preference SQL

Preference SQL ist eine Spracherweiterung von SQL, die es zusätzlich zur gewohnten Anfragefunktionalität erlaubt, Präferenzen für die Anfrageformulierung und -bearbeitung

heranzuziehen ([KK01], [Kie02]). Die Preference SQL Technologie ist mittlerweile kommerziell verfügbar (s. http://www.preference.de). Das Hauptanwendungsgebiet für Preference SQL liegt im Bereich des E-Shopping. Eine spezielle Preference-SQL-Unterstützung existiert beispielsweise für Intershop-Anwendungen.

Die Grundidee von Preference SQL besteht darin, Präferenzen als strikte partielle Ordnungen auf den vorliegenden Werten zu betrachten. Preference SQL unterstützt verschiedene Formen von Präferenzen. Im Folgenden werden zuerst die verfügbaren Basispräferenzen vorgestellt, anschließend dann die Verknüpfungsmöglichkeiten, die bei der Kombination von mehreren Präferenzen bestehen.

Für Attribute mit numerischen Werten existieren folgende Arten von Basispräferenzen:

- AROUND-Präferenz: Mit der AROUND-Präferenz kann der gewünschte Wert angegeben werden. Beispielsweise soll der Preis eines gesuchten Gebrauchtwagens etwa bei 20000 Euro liegen: `AROUND(Preis, 20000).`
- BETWEEN-Präferenz: Bei der BETWEEN-Präferenz liegen die gewünschten Werte in einem Intervall. Beispielsweise soll der gesuchte Gebrauchtwaren eine Laufleistung zwischen 20000 und 30000 Kilometern haben: `BETWEEN(Kilometerstand, [20000,30000]).`
- LOWEST/HIGHEST-Präferenz: Der gewünschte Wert sollte so klein bzw. so groß wie möglich sein. Beispielsweise soll der gesuchte Gebrauchtwagen soviel PS wie möglich haben: `HIGHEST(PS).`

Für nicht numerische Werte müssen die strikten partiellen Ordnungen explizit auf den Werten definiert werden. Dazu stehen folgende Präferenzen zur Verfügung:

- POS-Präferenz: Bei der POS-Präferenz wird eine Menge von positiven Werten angegeben. Der gewünschte Wert soll in dieser Menge sein. Die Werte, die nicht in dieser Menge sind, werden als schlechter angesehen. Die entstehende strikte partielle Ordnung besteht sozusagen aus zwei Stufen, den positiven und den übrigen Werten. Beispielsweise soll ein Fahrzeug mit Automatikgetriebe bevorzugt werden: POS(Getriebe, {Automatik}).
- NEG-Präferenz: Bei der NEG-Präferenz wird eine Menge von negativen Werten angegeben. Alle Werte, die nicht in dieser Menge sind, werden als besser angesehen als die angegebenen Werte. Die entstehende strikte partielle Ordnung hat ebenfalls zwei Stufen, wobei hier die untere Ebene explizit angegeben wird. Beispielsweise soll möglichst kein Ferrari gekauft werden: NEG(Marke, {Ferrari}).
- POS/NEG-Präferenz: Bei der POS/NEG-Präferenz werden zwei Mengen angegeben: eine Menge mit positiven und eine Menge mit negativen Werten. Die Werte in der Positivmenge sind allen anderen Werten vorzuziehen. Sind keine positiven Werte

verfügbar, werden alle Werte, die nicht in der Negativmenge aufgeführt sind, als besser angesehen als eben jene negativen Werte. Durch die POS/NEG-Präferenz wird eine strikte partielle Ordnung über drei Ebenen aufgebaut, wobei die Werte der obersten und der untersten Ebene explizit angegeben werden. Beispielsweise werden die Farben Schwarz und Rot bevorzugt, allerdings besteht eine Abneigung gegen die Farbe Grau: POS/NEG(Farbe, {schwarz, rot};{grau}).

- POS/POS-Präferenz: Bei der POS/POS-Präferenz werden ebenfalls zwei Mengen angegeben. Die Werte der Menge POS1 werden als besser angesehen als die Werte der Menge POS2. Die Werte der POS2 Menge sind allerdings immer noch besser zu bewerten als alle übrigen Werte. Hier wird wiederum eine dreistufige strikte partielle Ordnung aufgebaut, wobei die Werte der oberen beiden Ebenen explizit angegeben werden. Beispielsweise soll das gesuchte Auto am liebsten ein Cabrio sein. Ein Roadster wäre aber immer noch besser als alle übrigen Typen von Autos. (Typ, {Cabrio};{Roadster}).
- EXPLICIT-Präferenz: Mit der EXPLICIT-Präferenz kann mit den angegebenen Werten ein Besser-Als-Graph aufgebaut werden, indem seine Kanten explizit beschrieben werden. Dieser Graph kann im Gegensatz zu den vorangegangenen Präferenzen mehr als drei Ebenen enthalten, muss allerdings alle Werte auch explizit nennen. Z.B. EXPLICIT(Farbe, {schwarz, rot}{rot, gelb}{gelb, grün}{gelb, weiß} {grün, grau} {weiß, grau}).

Zur Verknüpfung mehrerer Präferenzen bietet Preference SQL zwei Möglichkeiten an:

- Pareto-Präferenz: Eine Pareto-Präferenz P setzt sich aus zwei (oder mehreren) gleich wichtigen Präferenzen zusammen. Ein Ergebnis e1 ist genau dann besser als ein anderes Ergebnis e2, wenn e1 eine Präferenz besser erfüllt als e2 und bzgl. aller anderer Präferenzen mindestens genauso gut abschneidet wie e2.
- Priorisierte Präferenz: Eine priorisierte Präferenz besteht aus zwei Präferenzen, wobei Präferenz P1 als wichtiger betrachtet wird als Präferenz P2. P1 wird zuerst zur Bewertung von Ergebnissen herangezogen. Ein Ergebnis e1 ist genau dann besser als ein anderes Ergebnis e2, wenn es entweder im Hinblick auf P1 besser abschneidet als e2 oder wenn e1 und e2 bzgl. P1 gleich bewertet werden und e1 die Präferenz P2 besser erfüllt als e2.

Die Semantik von Preference SQL wird als *BMO-Semantik* bezeichnet, d.h. *B*est *M*atches *O*nly. Das Ziel beim Einsatz von Preference SQL ist es, dass der Benutzer auf der einen Seite nicht mit einer Menge von Ergebnissen überschüttet wird, auf der anderen Seite aber auch keine leeren Ergebnismengen entstehen, wie es sehr leicht bei der Angabe von zu vielen harten Bedingungen geschehen kann. Das Ergebnis einer Preference-SQL-Anfrage wird nun folgendermaßen berechnet. Zuerst werden die Ergebnisse bestimmt, die die harten

Bedingungen erfüllen. Unter diesen Ergebnissen können nun aufgrund der formulierten Präferenzen partielle Ordnungen gebildet werden. Sofern es perfekte Ergebnisse gibt, d.h. Ergebnisse, die alle Präferenzen optimal erfüllen, werden nur diese Ergebnisse zurückgegeben. Wenn es keine perfekten Ergebnisse gibt, werden alle Ergebnisse zurückgegeben, zu denen es keine besseren Ergebnisse gibt (vgl. Abbildung 6.1).

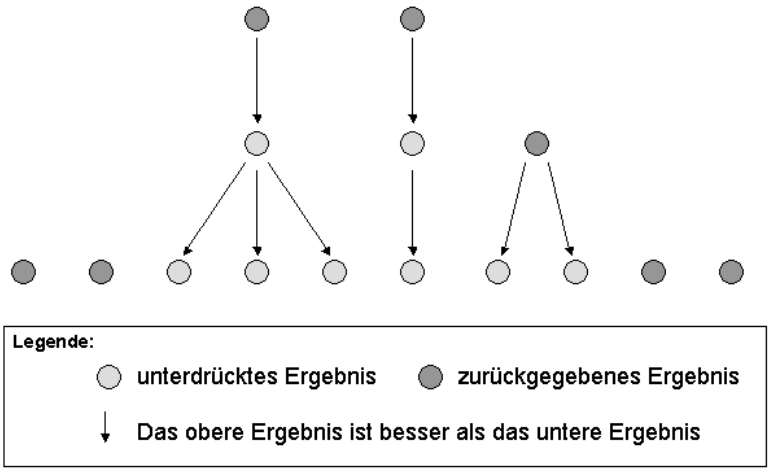


Abbildung 6.1: Partielle Ordnung und Markierung der zurückgegebenen Ergebnisse

Für den Umgang mit den in Preference SQL unterstützten Präferenzen wurde die Syntax der SELECT-Anweisung folgendermaßen erweitert.

```
SELECT          <selection>
FROM            <table_references>
WHERE           <where-conditions>
PREFERRING      <soft-conditions>
GROUPING        <attribute_list>
BUT ONLY        <but_only_condition>
ORDER BY        <attribute_list>
```

In der PREFERRING-Klausel werden die Präferenzen angegeben. Eine Pareto-Präferenz wird durch `AND` gebildet, eine priorisierte Präferenz wird durch das Schlüsselwort `CASCADE` kenntlich gemacht. Darüber hinaus sind spezielle Ausschlusskriterien formulierbar (BUT ONLY-Klausel). Ein solches Kriterium ist beispielsweise für Präferenzen auf numerischen Werten interessant, falls zusätzlich zu den bevorzugten Werten noch eine maximal tolerierte Abweichung angeben werden soll. Ohne dieses Ausschlusskriterium würden eventuell zu viele Ergebnisse präsentiert werden. Anstelle der Standard-SQL-Gruppierung ist bei Preference SQL die Gruppierung basierend auf Präferenzen möglich (GROUPING-Klausel). Dabei werden die Gruppen wie bei SQL auch aufgrund von gleichen Werten gebildet. Die angegebenen Präferenzen werden dann aber für jede einzelne Gruppe von Ergebnissen

ausgewertet, d.h. jede Gruppe enthält schließlich nur solche Ergebnisse, die von keinem anderen Gruppenmitglied übertroffen werden.

6.5.2 Preference XPath

Preference XPath [KHFH01] ist stark verwandt mit Preference SQL. Preference XPath wurde von derselben Forschergruppe entwickelt wie Preference SQL. Genau genommen wurden die Konzepte von Preference SQL für die Anwendung in XML-Anfragesprachen übertragen. Dazu ist XPath so erweitert worden, dass auch weiche Selektionsbedingungen bei der Formulierung von Pfadausdrücken verwendet werden können. Preference XPath unterstützt dieselben Arten von Präferenzen wie Preference SQL: die AROUND-Präferenz, die BETWEEN-Präferenz, die HIGHEST/LOWEST-Präferenz, die POS- und NEG-Präferenzen sowie die EXPLICIT-Präferenz. Es bestehen auch dieselben Kombinationsformen von Präferenzen, die Pareto-Präferenz und die priorisierte Präferenz.

Harte Selektionskriterien werden bei XPath innerhalb von eckigen Klammern, d.h. `[harte Selektionsbedingung]`, angegeben. Die Syntax von Preference XPath wurde so erweitert, dass die Präferenzen innerhalb von eckigen Klammern mit einem #-Zusatz angegeben werden, d.h. `#[weiche Selektionsbedingung]#`. Als weiche Selektionsbedingungen können alle unterstützten Formen und Kombinationen von Präferenzen verwendet werden. Wir wollen im Folgenden eine Beispiel-Anfrage in XQuery angeben, die Pfadausdrücke mit Präferenzen enthält.

```
<RESULT>
   FOR $a IN DISTINCT document("books.xml")//author
                              #[(text()) in ("Krisham", "Baker")]#
   RETURN
      <BooksByAuthor>
      <Author> $a/lastname/text() </Author>
      (FOR $b IN document("books.xml")//book[author=$a])
      RETURN
         $b/title #[(text()) in ("SECRET") and (text()) in ("WORLD")]#)
      </BooksByAuthor> SORTBY(Author)
</RESULT>
```

In diesem Beispiel wird nach den bevorzugten Autoren „Krisham“ oder „Baker“ gesucht. Da Preference XPath dieselbe BMO-Semantik unterliegt wie Preference SQL, werden hier entweder alle vorhandenen Bücher betrachtet, nämlich genau dann, wenn im gegebenen Dokumentbestand keine Bücher der beiden Autoren vorhanden sind. Wenn Bücher von einem oder beiden dieser Autoren im Dokumentbestand existieren, werden im Folgenden nur Bücher dieser beiden Autoren betrachtet. Für jeden ausgewählten Knoten wird nun eine Ausgabe produziert. Dabei wird jeweils der Nachname des Autors ausgegeben. Dazu werden alle Buchtitel von diesem Autor ausgegeben, bevorzugt aber diejenigen Titel, die die Worte „World“ und „Secret“ enthalten. Die BMO-Semantik bedeutet auch hier, dass im Erfolgsfall

nur die bevorzugten Titel ausgegeben werden, ansonsten aber alle Titel vom jeweiligen Autor aufgeführt werden.

Für Preference XPath existiert eine prototypische Implementierung, die auf dem kommerziellen XML-Datenbanksystem Tamino (von der Software AG) basiert.

6.5.3 Fazit

Ziel der Erweiterung von MLS-QL um weiche Bedingungen ist die Unterstützung von solchen Präferenzen, die im Bereich der Literaturrecherche häufig verwendet werden (vgl. Benutzerstudien) oder sinnvoll erscheinen. D.h. es sollen nicht beliebige Arten von Präferenzen in MLS-QL vorgesehen werden, sondern nur solche, die für die vorhandenen Attribute des Domänenmodells auch Sinn machen.

Die unterschiedlichen Arten von Basispräferenzen von Preference SQL bzw. XPath sind von den Wertebereichen der Attribute abhängig. Dabei wird zwischen numerischen und alphanumerischen Werten unterschieden. Auch das vorliegende Domänenmodell enthält Attribute mit numerischen und alphanumerischen Werten. Attribute mit numerischen Werten, die sich für die Formulierung von Präferenzen anbieten, sind beispielsweise Jahr, Auflage, Seitenzahl, Lieferpreis oder Lieferzeit. Auf diesen Attributen lassen sich die AROUND-, die BETWEEN-, und die HIGHEST/LOWEST-Präferenzen sinnvoll anwenden: `BETWEEN(Jahr, [2000, 2003]),LOWEST(Lieferzeit)` oder `AROUND(Seitenzahl, 150)`.

Bei alphanumerischen Attributen kann noch weiter unterschieden werden, nämlich ob das Attribut einen Wert aus einer fest vorgegebenen Menge an Werten enthält, wie beispielsweise bei Sprache, Publikationstyp oder Lieferart, oder ob das Attribut beliebige Werte enthalten kann, wie beispielsweise bei Titel, Autor, Verlag oder Schlagwort. Durch die verschiedenen POS/NEG-Präferenzen können beliebige lieber-als-Beziehungen angegeben werden. Diese Beziehungen können sich über mehrere Stufen erstrecken und können jeweils eine Stufe beinhalten, deren Werte nicht explizit angegeben werden müssen und die sozusagen stellvertretend für alle übrigen, nicht explizit genannten Werte stehen kann. Z.B. `POS/POS(Sprache,{de};{en})` oder `POS/NEG(Lieferart,{Ausleihe,Ansicht,Online, Fax};{Kauf})`. Allerdings haben bei dieser Vorgehensweise alle Stufen sozusagen denselben Abstand. An dieser Stelle wäre durchaus eine flexiblere Spezifikationsmöglichkeit für Abstände wünschenswert, beispielsweise durch die Vergabe von Gewichten.

Bei den Verknüpfungen von Basispräferenzen werden bei Preference SQL bzw. XPath zwei Varianten vorgesehen, nämlich dass zwei Präferenzen gleich wichtig sind und dass eine Präferenz wichtiger als eine andere Präferenz ist. Auch für die Verknüpfungen von Präferenzen sind flexiblere Kombinationsmöglichkeiten denkbar und wünschenswert, da es in der Tat wichtigere Präferenzen gibt, diese aber nicht unbedingt die übrigen Präferenzen so deutlich dominieren müssen. Wenn beispielsweise vier unterschiedlich wichtige Präferenzen angegeben werden, kann ein Dokument, welches die drei unwichtigeren Präferenzen erfüllt,

für den Benutzer relevanter sein als ein Dokument, welches lediglich die wichtigste Präferenz erfüllt. Daher sollten auch bei den Kombinationen mehrerer Präferenzen flexiblere Gestaltungsmöglichkeiten vorgesehen werden, beispielsweise ebenfalls über die Angabe von Gewichten.

Am meisten Probleme bei der Übertragung des vorgestellten Präferenzen-Konzepts auf das Literaturszenario bereitet allerdings die BMO-Semantik. In elektronischen Produktkatalogen mag es eine angemessene Vorgehensweise sein, diejenigen Produkte aus dem Ergebnis auszublenden, zu denen es gemäß der angegebenen Präferenzen ein besseres Produkt gibt. Bei der Literaturrecherche kann die Bewertung der Ergebnisse nicht ausschließlich anhand der angegebenen Kriterien erfolgen. Die Schlussfolgerung, dass ein Dokument, welches die Präferenzen etwas schlechter als ein anderes erfüllt, für den Benutzer nicht interessant ist und daher weggelassen werden kann, kann nicht gezogen werden. Bei der Literaturrecherche sind verschiedene Dokumente nicht direkt vergleichbar, hier entscheidet letztendlich immer der Benutzer aufgrund des konkreten Titels oder der Autoren über die Relevanz eines Dokuments. Auch ein gefundenes „optimales“ Buch soll nicht dazu führen, dass dem Benutzer keine weiteren Alternativen angezeigt werden.

Ein Beispiel soll die BMO-Semantik und die partiellen Ordnungen, die sich aufgrund der angegebenen Präferenzen ergeben, noch einmal verdeutlichen.

Beispiel 6.1: Gegeben ist eine Anfrage nach Büchern, in deren Titel das Wort Java vorkommt. Der Benutzer ist besonders interessiert an einem möglichst schnell verfügbaren Buch, dieses soll zudem auch möglichst aktuell sein und lieber in deutscher als in englischer Sprache geschrieben sein.

Mit den Mitteln mit Preference SQL bzw. Preference XPath können diese Präferenzen folgendermaßen formuliert werden:

```
LOWEST (Lieferzeit) CASCADE (HIGHEST(Jahr) AND POS/POS(Sprache,{de};{en}))
```

Abbildung 6.2 zeigt einen Teil der Ergebnismenge. Dabei wurden drei Quellen (UBKA, BLB und AMAZON) parallel befragt. Die ausleihbaren Bücher der UBKA sind in einer Stunde abholbereit, die ausleihbaren Bücher der BLB können in zwei Stunden abgeholt werden und die Lieferung der Bücher von AMAZON benötigt zwei bis drei Tage. Längere Wartezeiten treten auf, wenn ein Buch in einer Bibliothek gerade verliehen ist und keine anderen Beschaffungsalternativen bestehen. Zwischen den Büchern sind die partiellen Ordnungen gemäß der formulierten Präferenzen notiert. Die Präferenz bzgl. der kürzesten Lieferzeit ist dominant. Dadurch können die Dokumente in Klassen eingeteilt werden, wobei jeweils alle Dokumente der oberen Klasse besser bewertet werden als sämtliche Dokumente der darunter liegenden Klassen. Innerhalb von einer Klasse können weitere partielle Ordnungen vorliegen. Dabei ist ein Dokument genau dann besser als ein anderes Klassenmitglied, wenn es mindestens eine Präferenz besser erfüllt und dabei bei den restlichen Präferenzen gleich oder

besser als das andere Klassenmitglied abschneidet. Die BMO-Semantik gibt lediglich das oberste Dokument zurück und argumentiert, dass der Benutzer nur dieses Dokument zu sehen braucht, da alle anderen Dokumente gemäß seiner Präferenzen schlechter abschneiden als dieses.

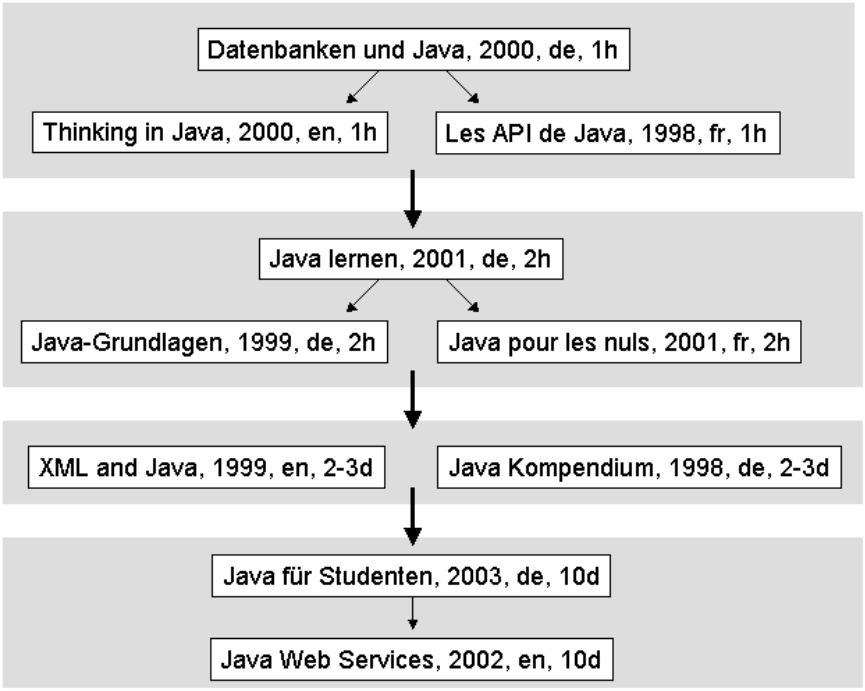


Abbildung 6.2: Partielle Ordnungsrelation zwischen den Ergebnissen der Beispielanfrage

Für das vorliegende Literaturszenario ist die BMO-Semantik nicht geeignet. Wünschenswert wäre vielmehr, dass prinzipiell alle gefundenen Dokumente auch angezeigt werden. Dabei soll jedes Dokument einen Relevanzwert entsprechend der gegebenen Präferenzen erhalten, so dass die Ergebnismenge beispielsweise auch nach der Relevanz der Dokumente sortiert werden kann. Die Dokumente in Abbildung 6.2 machen auch noch einmal die thematischen Unterschiede deutlich, die zwischen den gefundenen Dokumenten bestehen können. Hier kann nur der Benutzer selbst entscheiden, an welchen Dokumenten er (nicht) interessiert ist. Will der Benutzer tatsächlich nur wenige Dokumente präsentiert bekommen, kann er die Größe der Ergebnismenge auch separat beschränken (vgl. Abschnitt 6.4, `TOP(k)`). Dann entscheidet die gewählte Sortierreihenfolge, welche Dokumente angezeigt werden. Die Möglichkeiten der Sortierung müssen in MLS-QL also noch um die Relevanz hinsichtlich angegebener Präferenzen erweitert werden.

6.6 Erweiterung der Anfragesprache um Präferenzen

In diesem Abschnitt wird die in 6.4 spezifizierte Anfragesprache MLS-QL um Präferenzen erweitert. Die Angabe von Präferenzen soll die Menge der gefundenen Ergebnisse nicht verändern, d.h. die Anzahl der Treffer bleibt gleich, es sei denn, der Benutzer hat die Größe der gewünschten Ergebnismenge explizit beschränkt. Präferenzen geben Hinweise auf die Relevanz von Dokumenten. Dabei sollen sowohl die Basispräferenzen als auch die Verknüpfungen der Basispräferenzen flexibel spezifiziert werden können.

Werden mehrere Präferenzen angegeben, so können diese von unterschiedlicher Wichtigkeit für den Benutzer sein. Hier sollen nicht nur gleich wichtige und dominante Präferenzen zugelassen werden, sondern es soll auch die flexible Gewichtung von Präferenzen unterstützt werden. Jede $Präferenz_i$ kann dabei mit einem Gewicht w_i versehen werden, wobei gelten soll: $0 < w_i <= 1$. Die Standardbelegung von w_i ist 1.

Dafür wird die Syntax von MLS-QL folgendermaßen erweitert:

```
  SELECT          [TOP(k)] <projektionsausdruck>
  FROM            <quellenauswahl>
  WHERE           <selektionsbedingung>
[ DUPLICATES BY   <attributname> [ USING <vergleichsoperator>
                (, <attributname> [ USING <vergleichsoperator> ] )*      ]
[ PREFERRING      [w1] Präferenz1 ( AND [wi] Präferenzi )*               ]
[ GROUP BY        <attributname> [ SOME | ALL ]
                (, <attributname> [ SOME | ALL ] )*                    ]
[ ORDER BY        <sortierung>                                           ] ;
```

Bei der Unterstützung von Basispräferenzen soll ebenfalls mehr Flexibilität erlaubt werden als es die partiellen lieber-als-Beziehungen von Preference SQL bzw. XPath zulassen. Dazu soll es beispielsweise bei einer Präferenz von möglichst aktuellen Büchern erlaubt sein, den verschiedenen Jahreszahlen explizit unterschiedliche Gewichte zuzuordnen.

Durch die Einführung einer Gewichtung kann auch die Spezifikation von Präferenzen vereinheitlicht werden, so dass nicht unterschieden werden muss, ob sich eine Präferenz auf ein numerisches oder ein alphanumerisches Attribut bezieht. Eine Präferenz wird nicht direkt in der Anfrage angegeben, sondern zuvor in einem separaten Schritt definiert. Dabei erhält eine $Präferenz_i$ einen eindeutigen Namen, mit welchem sie dann innerhalb einer Anfrage aufgerufen werden kann. Eine $Präferenz_i$ bezieht sich auf genau ein $Attribut_i$.

```
CREATE PREFERENCE Präferenzi (Attributi) AS {
    vi1: ( Wert | Wertebereich | Wortmenge );
[ ( vij: ( Wert | Wertebereich | Wortmenge ); )*          ]
[   vimi: ELSE;                                           ]
};
```

Dabei ist die Angabe eines Wertes bzw. eines Wertebereichs für numerische Attribute, die Angabe einer Wortmenge für alphanumerische Attribute vorgesehen. Für die Gewichte v_{ij} soll

gelten: -1 <= v_{ij} <= 1. Durch die Angabe von negativen Gewichten können beispielsweise auch Abneigungen für gewisse Werte formuliert werden. In einer Präferenz kann maximal einmal auf die nicht explizit genannten Werte Bezug genommen werden, nämlich durch das Schlüsselwort `ELSE`.

Für die Semantik der definierten MLS-QL-Präferenzen gilt folgendes: Die Präferenzen dienen dazu, Hinweise über die Relevanz (vgl. Definition 4.1) eines Dokuments zu bekommen. Mit Relevanz wird nicht die thematische Eignung eines Dokuments bezeichnet, sondern die Relevanz im Hinblick auf die Entscheidungsfindung des Benutzers, d.h. inwieweit das Dokument die Informationen enthält bzw. die Kriterien erfüllt, die der Benutzer bei der Auswahl eines Dokuments heranziehen möchte. Für jedes Dokument kann nun ein Relevanzwert berechnet werden, der den Grad der Erfüllung der genannten Präferenzen widerspiegelt.

$$\text{Relevanz von Dokument D} = \sum_{i=1}^{n} \sum_{j=1}^{mi} w_i * v_{ij} * \texttt{liegt\_in}(D,i,j)$$

Dabei ist `liegt_in(D,i,j)` eine boolesche Funktion. Sie hat den Wert 1, wenn das zur $Präferenz_i$ zugehörige $Attribut_i$ des Dokuments D in den Wertebereich von v_{ij} fällt, andernfalls liefert die Funktion den Wert 0. Aufgrund von dieser Relevanzberechnung kann nun beispielsweise eine geeignete Ergebnisreihenfolge bestimmt werden. Dazu muss die Anfragesprache MLS-QL um ein zusätzliches Sortierkriterium und ein entsprechendes Schlüsselwort (`RELEVANCE`) ergänzt werden.

Gegeben sei noch einmal Beispiel 6.1 aus dem vorangegangenen Abschnitt: Ein Benutzer fragt nach Büchern, in deren Titel das Wort Java vorkommt. Der Benutzer ist besonders interessiert an einem möglichst schnell verfügbaren Buch, dieses soll zudem auch möglichst aktuell sein und lieber in deutscher als in englischer Sprache geschrieben sein.

Mit den MLS-QL-Präferenzen können die Vorlieben des Benutzers nun differenzierter ausgedrückt und flexibler miteinander verknüpft werden. Die Vorliebe für möglichst schnell verfügbare Bücher könnte folgendermaßen spezifiziert werden: Am liebsten hat der Benutzer Bücher, die er innerhalb von einer Stunde (h) beziehen kann. Schon, wenn er bis zu zwei Stunden warten muss, ist ihm das nicht so angenehm. Allerdings wird eine Lieferzeit von zwei bis drei Tagen (d) immer noch geschätzt. Als negativ werden Bücher bewertet, auf die der Benutzer länger als 2 Wochen (w) warten müsste.

```
CREATE PREFERENCE möglichst_schnell_verfügbar(Lieferzeit) AS {
   1,0: [0h-1h];
   0,8: (1h-2h];
   0,6: (2h-3d];
  -1,0: (14d-∞);
};
```

Die Vorliebe für aktuelle Bücher könnte folgendermaßen angegeben werden: Der Benutzer bevorzugt Bücher der letzten fünf Jahre, wobei das Interesse dabei kontinuierlich abnimmt.

```
CREATE PREFERENCE möglichst_aktuell(Jahr) AS {
   1,0: 2003;
   0,8: 2002;
   0,6: 2001;
   0,4: 2000;
   0,2: 1999;
};
```

Und schließlich kann noch eine Präferenz für die Sprache von Büchern angegeben werden. Der Benutzer bevorzugt zwar deutsche Bücher, allerdings ist für ihn der Mehrwert im Vergleich zu englischen Büchern nicht allzu groß.

```
CREATE PREFERENCE lieber_deutsch_als_englisch(Sprache) AS {
   1,0: {"de"};
   0,7: {"en"};
};
```

Eine MLS-QL-Anfrage könnte nun folgendermaßen aussehen:

```
   SELECT         Originaltitel, Jahr, Sprache, Lieferzeit
   FROM           UBKA, BLB, AMAZON
   WHERE          Originaltitel LIKE "%Java%"
   PREFERRING     1,5 möglichst_schnell_verfügbar(Lieferzeit)
   AND            1,0 möglichst_aktuell(Jahr)
   AND            1,0 lieber_deutsch_als_englisch(Sprache)
   ORDER BY       RELEVANCE;
```

In der ORDER-BY-Klausel kann das neu eingeführte Schlüsselwort `RELEVANCE` verwendet werden, um eine Ordnung der Ergebnisse aufgrund ihrer Relevanz zu den Präferenzen zu ermöglichen. Das Schlüsselwort `RELEVANCE` kann nur in Kombination mit der Angabe von Präferenzen verwendet werden, hat aber bei der Sortierung von Gruppen keine Wirkung.

Abbildung 6.3 greift noch einmal die partielle Ordnung auf, die durch die Präferenzen in Preference SQL bzw. XPath entstanden sind. Die Dokumente wurden jeweils mit Relevanzwerten versehen, die sich aufgrund der MLS-QL Präferenzen ergeben. Der rechte Teil der Abbildung zeigt die Ausgabereihenfolge der Dokumente für die oben angegebene MLS-QL-Anfrage.

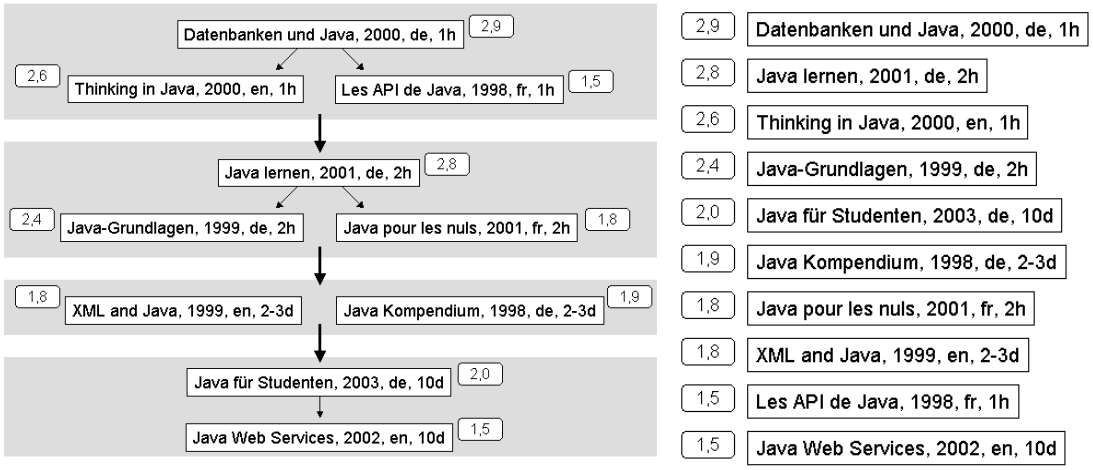


Abbildung 6.3: Vergleich der Ergebnisse mit Preference SQL/XPath (li.) und mit MLS-QL (re.)

6.7 Resümee

In diesem Kapitel wurden zunächst die Anforderungen an eine geeignete Anfragefunktionalität geklärt. Daraus wurde für den Lösungsansatz eine zweistufige Vorgehensweise abgeleitet, um die Summe der Anforderungen erfüllen zu können.

Im vorliegenden Kapitel wurde im ersten Schritt eine hinreichend mächtige Anfragesprache namens MLS-QL entwickelt. MLS-QL orientiert sich stark an SQL, wurde aber für die speziellen Bedürfnisse bei einer Meta-Literaturrecherche angepasst bzw. erweitert. Die Erweiterungen betreffen zum einen den Umgang mit mehrwertigen Attributen, zum anderen die flexiblere Unterstützung im Hinblick auf Gruppenbildung und Duplikaterkennung. Darüber hinaus wurde MLS-QL um Präferenzen ergänzt, die es dem Benutzer erlauben, persönliche Vorlieben und Entscheidungskriterien zur Bewertung von Dokumenten heranzuziehen. Durch den Einsatz von Gewichten lassen sich die MLS-QL-Präferenzen beliebig gestalten und flexibel kombinieren. Die Verwendung von Präferenzen hat dabei keinen Einfluss auf die Größe der Ergebnismenge, sie gibt lediglich Hinweise auf die Relevanz von Dokumenten, die beispielsweise für eine bessere Sortierreihenfolge oder eine geeignete Präsentation an der Benutzungsschnittstelle ausgenutzt werden können.

Im folgenden Kapitel erfolgt nun der zweite Schritt des Lösungsansatzes: Für die spezifizierte Anfragesprache MLS-QL soll nun eine geeignete Umsetzung der Anfragefunktionalität an der Benutzungsschnittstelle entwickelt werden.

7 Konzeption und Umsetzung einer Rechercheschnittstelle

7.1 Überblick

Im vorangegangenen Kapitel wurde eine Anfragesprache namens MLS-QL definiert, die durch beliebig kombinierbare harte und weiche Bedingungen sowie durch flexible Gruppierungs- und Duplikaterkennungsmöglichkeiten hinreichend viele Ausdrucksmittel für die Formulierung von Anfragen bei einer Literaturrecherche zur Verfügung stellt. Damit kann der erste Teil der in Abschnitt 6.2 genannten Anforderungen als erfüllt betrachtet werden. Der zweite Teil der Anforderungen, der die einfache und intuitive Benutzbarkeit eines Meta-Recherchesystems betrifft, ist Gegenstand des vorliegenden Kapitels (Abschnitt 7.2).

Der vorgeschlagene Lösungsansatz beschäftigt sich zunächst mit der Reduzierung der Komplexität der Anfrageformulierung für den Benutzer. Dazu wird das Konzept der generischen Suchmuster entwickelt (Abschnitt 7.3). Dabei wird eine Anfrage in einen aktuellen thematischen Bezug und in häufiger auftretende wiederverwendbare Such- bzw. Entscheidungsstrategien zerlegt. Für jeden Benutzer wird ein individuelles Repertoire an generischen Suchmustern vorgesehen.

Anschließend wird eine geeignete Rechercheschnittstelle entworfen, die mit den bekannten und damit dem Benutzer auch relativ vertrauten WIMP-Gestaltungsmitteln realisiert wird (Abschnitt 7.4). Über diese Elemente werden die einzelnen Anfragefunktionalitäten sowie die Ergebnispräsentation einfach und intuitiv für den Benutzer handhabbar gemacht.

7.2 Anforderungen und Lösungsansatz

Dieses Kapitel beschäftigt sich mit dem Teil der in Abschnitt 6.2 formulierten Anforderungen, die durch eine geeignete Gestaltung der Benutzungsschnittstelle erfüllt werden können. Dazu soll eine einfache und intuitive Benutzerinteraktion gewährleistet werden, die den menschlichen Fähigkeiten, Fertigkeiten und Grenzen Rechnung trägt. D.h. es sollen – wenn möglich – Erinnerungshilfen gegeben und keine zu hohen Erwartungen an den Benutzer gestellt werden. Dabei sollen die Rechercheergebnisse einer Anfrage möglichst schnell angezeigt werden, so dass für den Benutzer keine langen Wartezeiten entstehen.

Die Betrachtung von verwandten Arbeiten im Rahmen von Kapitel 3 hat zum einen zusätzliche Anforderungen, aber auch bereits bewährte Realisierungstechniken identifizieren können. Der Gestaltungsrahmen formuliert eine Reihe von Kriterien, die von benutzerfreundlichen Rechercheschnittstellen erfüllt werden müssen. Dazu werden auch konkrete

Anhaltspunkte und Vorschläge für eine geeignete Realisierung gegeben. Das SenseMaker-Projekt belegt zum einen die Notwendigkeit und den Nutzen von interaktiven Nachbearbeitungsoperatoren für die Begutachtung großer Ergebnismengen und kann zum anderen die Präsentation der Ergebnisse durch den Einsatz von Hi-Cites wesentlich übersichtlicher gestalten. Der DLITE-Ansatz verdeutlicht die Intuitivität und Klarheit des Drag&Drop-Prinzips. Der vorgeschlagene Lösungsansatz soll zur Erfüllung der Anforderungen existierende und bewährte Realisierungsansätze berücksichtigen und in geeigneter Weise aufgreifen.

Die spezielle Herausforderung des Lösungsansatzes besteht in der vorliegenden Arbeit darin, die offensichtliche Diskrepanz zwischen der ausdrucksmächtigen Anfragesprache MLS-QL und einer einfachen und intuitiven Benutzerinteraktion zu überbrücken. Benutzerstudien haben gezeigt, dass Anfragen von Benutzern oft nur aus wenigen und einfach verknüpften Angaben bestehen. Um auf der anderen Seite bei Meta-Recherchesystemen handhabbare bzw. zu bewältigende Ergebnismengen zu erzielen, sind hinreichend präzise und damit eben auch aufwendige Anfragespezifikationen notwendig. Der vorliegende Lösungsansatz versucht die Komplexität der Anfrageformulierung für den Benutzer durch die Konzeption und den Einsatz von generischen Suchmustern zu reduzieren. Generische Suchmuster können als Teile von Anfragen beschrieben werden, die häufiger auftreten bzw. in verschiedenen Zusammenhängen verwendet werden können. Sie spiegeln im Wesentlichen eine persönliche Such- bzw. Entscheidungsstrategie eines Benutzers wider.

Im weiteren Verlauf des Kapitels wird zunächst der konzeptuelle Kern des Lösungsansatzes, nämlich das Konzept der generischen Suchmuster, entwickelt und detailliert betrachtet (Abschnitt 7.3). Dazu werden eine Reihe von Definitionen benötigt sowie die Behandlung von möglichen Konflikten beim Verknüpfen eines generischen Suchmusters mit einer aktuellen Suchanfrage.

Im folgenden Abschnitt wird dann ein Realisierungskonzept und -vorschlag für eine geeignete Rechercheschnittstelle ausgearbeitet (Abschnitt 7.4). Neben einer Schnittstelle für die Anfrageformulierung und die Präsentation der Ergebnismenge wird auch eine geeignete Umsetzung für die zuvor konzipierten generischen Suchmuster erarbeitet.

7.3 Generische Suchmuster

In diesem Abschnitt wird das Konzept der generischen Suchmuster entworfen und konkretisiert. Die Umsetzung der generischen Suchmuster an der Benutzungsschnittstelle erfolgt im Rahmen von Abschnitt 7.4.

7.3.1 Motivation

Das zentrale Problem bei der Gestaltung eines Lösungsansatzes liegt in der Diskrepanz zwischen der notwendigen Mächtigkeit der MLS-QL-Anfragesprache und den Fähigkeiten

bzw. Fertigkeiten der Benutzer bei der Anfrageformulierung. Auf der einen Seite sind für eine effiziente Anfragebearbeitung und -optimierung hinreichend präzise Anfragen notwendig. Auf der anderen Seite haben die Transaction Log Analysen in Abschnitt 3.2.2 gezeigt, dass Benutzer durchschnittlich nur zwei bis drei Angaben bei einer Anfrage machen, aus welchen Gründen auch immer, z.B. aus Zeitmangel oder aufgrund mangelnder Suchfertigkeiten.

Die Grundidee des vorliegenden Lösungsansatzes besteht nun darin, wiederkehrende Teile von Benutzeranfragen zu identifizieren und zu separieren. Diese Teile können dann zu einem späteren Zeitpunkt wiederverwendet werden und dadurch die Komplexität der aktuellen Anfrageformulierung reduzieren.

Die Ausgangslage für diesen Ansatz bilden die in qualitativen Benutzerstudien ermittelten Informationsbedürfnisse der Benutzer. Studien wie [Wan97] haben gezeigt, dass Benutzer in der Regel nur wenige Entscheidungsstrategien und nur wenige Entscheidungskriterien anwenden. Am weitesten verbreitet ist die Entscheidungsstrategie, die ein Dokument aufgrund eines dominanten Attributs, d.h. einer dominanten Eigenschaft, auswählt. Benutzer entscheiden sich beispielsweise bei einer Menge von Dokumenten für das aktuellste (Entscheidungskriterium Erscheinungsjahr), das thematisch geeignetste (Entscheidungskriterium Thema) oder das am schnellsten verfügbare Dokument (Entscheidungskriterium Lieferzeit). Die zweithäufigste Entscheidungsstrategie berücksichtigt eine Kombination von Eigenschaften. Dabei werden vom Benutzer zwei oder mehr Entscheidungskriterien angewandt und die Dokumente gewählt, die in der Kombination dieser Eigenschaften am besten abschneiden.

Diese Arten von Entscheidungsstrategien und -kriterien werden von MLS-QL, insbesondere von den Präferenzen, den Gruppierungs- und Sortierungsoptionen, bereits in geeigneter Weise unterstützt. Die Attribute bzw. Eigenschaften, die im Rahmen von diesen Entscheidungskriterien verwendet werden, wurden bereits bei der Erstellung des Domänenmodells vorgestellt und berücksichtigt (vgl. Abschnitt 5.3.2). Dabei wurde deutlich, dass die thematisch orientierten Eigenschaften nur einen Bruchteil der Kriterien ausmachen, die von Benutzern zur Beurteilung von Dokumenten herangezogen werden.

Im Rahmen der Entscheidungsprozesse verwenden Benutzer in der Regel eine kleine Menge an Entscheidungskriterien, die sie auch immer wieder in unterschiedlichen Zusammenhängen verwenden. Die Menge der verwendeten Entscheidungskriterien und -strategien variiert nur wenig, beispielsweise dann, wenn bei Kollegen neue Anregungen entdeckt und aufgegriffen werden. Teilweise entwickelt sich auch die Suchfertigkeit des Benutzers weiter, so dass ausgefeiltere Entscheidungsstrategien bzw. -kriterien in das persönliche Repertoire hinzugenommen werden. Meist fallen dabei auch weniger geeignete oder veraltete Entscheidungsstrategien bzw. -kriterien aus dem regelmäßig genutzten Repertoire heraus, so dass der Umfang im Wesentlichen über die Zeit stabil bleibt.

Diese Beobachtungen liefern die wesentlichen Anhaltspunkte für die folgende Konzeption: Wir wollen jedem Benutzer ein persönliches Repertoire an Entscheidungsstrategien bzw.

-kriterien zuordnen. Dabei ist es nicht nötig, explizit zwischen Entscheidungsstrategie und -kriterium unterscheiden, da in MLS-QL deren Kombination gemeinsam ausgedrückt wird, indem verschiedene Bedingungen (harte und weiche) miteinander verknüpft werden. Bei diesen Bedingungen sollen keine thematischen Angaben gemacht werden, um die Wiederverwendbarkeit in verschiedenen Zusammenhängen zu gewährleisten. Für dieses Konzept wird im Folgenden der Begriff des *Generischen Suchmusters* verwendet. Generisch deutet an, dass die Bedingungen für sich alleine genommen noch nicht ausreichend sind, um als Anfrage zu gelten. Zusätzlich werden weitere Angaben vom Benutzer benötigt, nämlich die aktuelle Anfrage, die den jeweiligen thematischen Bezug enthalten sollte. Suchmuster ist nun der Begriff, unter dem Entscheidungsstrategie und -kriterium zusammengefasst werden. Der Begriff Suchmuster macht auch deutlich, dass die Bedingungen bereits zur Suche mit angegeben werden müssen und damit auch Auswirkungen auf die Anfragebearbeitung haben können. Die Bezeichnung als Muster soll andeuten, dass ein Muster in verschiedenen Kontexten auftauchen kann.

Dadurch setzt sich eine Rechercheanfrage aus einer aktuellen Benutzeranfrage, die einen thematischen Bezug beinhalten sollte, und einem generischen Suchmuster zusammen. Die aktuelle Anfrage werden wir im Folgenden auch als ad-hoc-Anfrage des Benutzers bezeichnen. Durch die Konzeption der generischen Suchmuster kann der Beobachtung Rechnung getragen werden, dass Benutzer bei ihren Anfragen oft nur zwei bis drei Angaben machen.

Die folgende Aufzählung soll noch einmal die wichtigsten Punkte beim Entwurf der generischen Suchmuster zusammenstellen:

- Jeder Benutzer besitzt ein persönliches Repertoire an generischen Suchmustern.
- Der Umfang eines persönlichen Repertoires ist meist recht überschaubar.
- Unterschiedliche Benutzer haben meist unterschiedliche generische Suchmuster in ihrem Repertoire.
- Der Benutzer kann sein persönliches Repertoire an generischen Suchmustern beliebig ergänzen oder verfeinern.
- Ein generisches Suchmuster kann vom Benutzer in verschiedenen Zusammenhängen verwendet werden.
- Eine Rechercheanfrage setzt sich aus einer ad-hoc-Anfrage (mit thematischen Bezug) und ggf. aus einem generischen Suchmuster zusammen.

Im folgenden Abschnitt wird das Konzept der generischen Suchmuster verfeinert und präzisiert.

7.3.2 Definition und Konzeption

Im vorangegangenen Abschnitt wurde das Konzept der generischen Suchmuster motiviert und bereits in groben Zügen umrissen. In diesem Abschnitt soll nun eine präzisere Definition eines generischen Suchmusters erfolgen, gerade auch im Hinblick auf die Umsetzung mit MLS-QL-Konstrukten. Dies soll dadurch geschehen, dass diejenigen MLS-QL-Konstrukte identifiziert werden, die zur Spezifikation eines generischen Suchmusters eingesetzt werden dürfen. Wir wollen zunächst mit einer allgemeinen Definition eines generischen Suchmusters beginnen und diese im weiteren Verlauf zu konkretisieren versuchen.

Definition 7.1: Ein *generisches Suchmuster* ist ein Teil einer Rechercheanfrage, der in verschiedenen Zusammenhängen, d.h. in verschiedenen Rechercheanfragen, wiederverwendet werden kann, für sich alleine genommen aber keine vollständige Rechercheanfrage darstellt.

Wir wollen zunächst den zweiten Teil der Definition betrachten. Im Rahmen der Spezifikation von MLS-QL (vgl. Abschnitt 6.6) wurde definiert, dass eine MLS-QL-Anfrage dann vollständig ist, wenn sie mindestens einen SELECT-, FROM- und WHERE-Teil enthält. Diese Definition ist im Grunde genommen noch zu allgemein gehalten, da keine Aussage über die Bedingungen gemacht wird, die in der WHERE-Klausel angegeben werden. Dadurch würde nämlich auch eine Anfrage mit einer immer wahren Bedingung (TRUE) oder eine Anfrage nach allen Büchern aus dem Jahr 2002 als vollständige Anfrage gelten. Solche Anfragen wären allerdings bei größeren Datenbeständen, wie sie bei Meta-Recherchesystemen zu erwarten sind, sicherlich nicht zielführend. Eine vollständige Rechercheanfrage im Literaturbereich sollte im Allgemeinen mindestens eine Bedingung (Stichwortangabe) an Titel, Autor oder Schlagwort beinhalten. Weiterhin sind auch Anfragen nach einer gegebenen ISBN als vollständig zu betrachten. Anfragen, die ausschließlich Bedingungen an die übrigen Attribute des Domänenmodells enthalten, können bei größeren Datenbeständen nicht sinnvoll verwendet werden.

Definition 7.2: Eine MLS-QL-Anfrage heißt *vollständig*, wenn sie eine SELECT-, eine FROM- und eine WHERE-Klausel aufweist, wobei in der WHERE-Klausel mindestens eine Bedingung an Titel, Autor, Schlagwort oder ISBN enthalten sein muss.

Im Folgenden wollen wir nun betrachten, welche Teile von Anfragen als generische Suchmuster verwendet werden können. Dazu wollen wir zwei Aspekte betrachten: den inhaltlichen und den formalen Aspekt. Der inhaltliche Aspekt beschäftigt sich damit, welche Attribute bei generischen Suchmustern verwendet werden können. Der formale Betrachtungsansatz greift die einzelnen Bestandteile einer MLS-QL-Anfrage auf und untersucht, welche MLS-QL-Konstrukte im Rahmen von generischen Suchmustern verwendet werden können.

Inhaltlicher Aspekt (Attribute des Domänenmodells). Wir wollen zunächst den inhaltlichen Aspekt einer Anfrage betrachten. Typische Attribute, die im Rahmen von generischen Suchmustern verwendet werden, sind sicherlich Jahr, Publikationstyp, Lieferzeit, Lieferformat, Lieferadresse oder Lieferkosten (Abschnitt 5.3.2). Wir müssen uns nun die Frage

stellen, ob die bei der Vollständigkeitsdefinition genannten Attribute ausgeschlossen werden können. Für das Titel-Attribut wäre beispielsweise ein generisches Suchmuster denkbar, das nach den Worten Überblick, Einführung oder den englischen Pendants dazu sucht. Das würde bedeuten, dass der Benutzer an Einführungs- oder Überblickswerken interessiert ist, sich dazu schon einmal die sinnvollen Begriffe (mit einer Oder-Verknüpfung) zusammengestellt hat und dieses Muster nun nur noch mit einem aktuellen Thema instanziieren muss. In ähnlicher Weise könnte der Benutzer eine Menge von etablierten oder renommierten Autoren angeben, deren Publikationen er bevorzugen würde. Auch bei Schlagwort ist beispielsweise eine allgemeine Angabe wie Informatik nicht unwichtig, um dadurch beispielsweise Bücher über die Insel Java oder über Fenster (Windows) auszuschließen. Eine Bedingung an das ISBN-Attribut kann sicherlich für generische Suchmuster ausgeschlossen werden, die anderen Attribute des Domänenmodells können allerdings prinzipiell sinnvoll verwendet werden.

Fazit: Ein generisches Suchmuster kann prinzipiell alle Attribute des Domänenmodells (mit Ausnahme von ISBN) betreffen.

Formaler Aspekt (Bestandteile einer MLS-QL-Anfrage). Für die Betrachtung des formalen Aspektes untersuchen wir die Konstrukte in MLS-QL, die im Rahmen von generischen Suchmustern verwendet werden können. Im Wesentlichen enthalten generische Suchmuster Angaben in den WHERE- (harte Bedingungen) und PREFERRING-Klauseln (weiche Bedingungen). Prinzipiell ist es aber auch möglich, persönliche Vorlieben im Hinblick auf die Duplikaterkennung (DUPLICATES), die Gruppierung (GROUP BY) und die Sortierung (ORDER BY) zu haben. Darüber hinaus kann ein Benutzer auch eine spezielle Attributmenge bzw. Trefferanzahl (SELECT) oder eine spezielle Quellenkombination (FROM) bevorzugen. Damit kann auch bei den MLS-QL-Klauseln keine prinzipielle Einschränkung getroffen werden. Es ist sicherlich empfehlenswert, bei der Spezifikation von generischen Suchmustern weiche Bedingungen zusammen mit einer RELEVANCE-Sortierung zu verwenden, da dadurch keine Ergebnisse ausgeschlossen, aber die relevantesten Ergebnisse zuerst präsentiert werden. Prinzipiell kann allerdings keine Einschränkung gemacht werden.

Fazit: Zur Spezifikation eines generischen Suchmusters können prinzipiell alle MLS-QL-Konstrukte verwendet werden.

Aufgrund der inhaltlichen und der formalen Betrachtungen kann keine weitere Eingrenzung eines generischen Suchmusters erfolgen. Zur Spezifikation können prinzipiell alle Attribute (bis auf ISBN) und alle Konstrukte von MLS-QL in sinnvoller Weise verwendet werden. Es kann also nicht automatisch per Definition festgestellt werden, ob eine vom Benutzer als generisches Suchmuster deklarierte Teilanfrage auch „korrekt“ ist, d.h. sinnvoll genutzt werden kann. Diese Korrektheitsprüfung muss somit dem Benutzer selbst überlassen bleiben.

Im nächsten Schritt wollen wir die Granularität eines generischen Suchmusters detaillierter betrachten. Im vorangegangenen Abschnitt haben wir vorläufig festgelegt, dass jeweils

maximal ein generisches Suchmuster mit einer ad-hoc-Anfrage kombiniert werden kann. Prinzipiell sind an dieser Stelle aber auch verschiedene Varianten denkbar. Zwei extreme Varianten können wie folgt umrissen werden.

Variante 1 (feine Granularität): Die Spezifikation eines generischen Suchmusters darf nur eine MLS-QL-Klausel enthalten, beispielsweise eine Bedingung (WHERE, ohne boolesche Verknüpfung), eine Präferenz (PREFERRING, ohne Kombination), eine Sortierreihenfolge (ORDER BY) oder eine bestimmte Auswahl von Diensten (FROM).

Diese Variante bedeutet entweder, dass die ad-hoc-Anfrage noch recht viele zusätzliche Angaben enthalten muss, was dem aktuellen Verhalten der Benutzer widersprechen würde, oder dass die Kombination mehrerer generischer Suchmuster zu einer ad-hoc-Anfrage zugelassen werden müsste. Die manuelle Kombination mehrerer generischer Suchmuster würde allerdings vermutlich nicht zur Vereinfachung und Klarheit auf Seiten des Benutzers beitragen.

Variante 2 (grobe Granularität): Ein generisches Suchmuster soll möglichst zu allen MLS-QL-Konstrukten Angaben enthalten, so dass der Benutzer in der ad-hoc-Anfrage nur noch das aktuelle Stichwort ergänzen muss.

Der Vorteil dieser Variante liegt auf der Hand. Bei der ad-hoc-Anfrage braucht der Benutzer nur wenige Angaben zu machen und nur ein generisches Suchmuster auszuwählen. Allerdings wird dadurch zwischen den einzelnen generischen Suchmustern eine erhebliche Redundanz entstehen, da für jede auch noch so gering abweichende Einstellung ein neues generisches Suchmuster spezifiziert werden muss. In diesem Fall muss der Benutzer bei der Auswahl eines passenden generischen Suchmusters sehr achtsam sein und auch gerade die kleinen Unterschiede zur Kenntnis zu nehmen.

Fazit: Auch was die Granularität der generischen Suchmuster betrifft, kann keine prinzipielle Regelung eingesetzt werden. Aus Gründen der Klarheit und Übersichtlichkeit ist es sicherlich sinnvoll, auszuschließen dass mehrere generische Suchmuster mit einer ad-hoc-Anfrage verknüpft werden können. Mit wie vielen und welchen Angaben ein generisches Suchmuster schließlich ausgestattet wird, kann jeder Benutzer individuell entscheiden.

Definition 7.3: Eine *Rechercheanfrage* besteht aus einer ad-hoc-Anfrage, die mit maximal einem generischen Suchmuster verknüpft werden kann. Insgesamt muss eine Rechercheanfrage eine vollständige MLS-QL-Anfrage darstellen.

Bei der Verknüpfung einer ad-hoc-Anfrage mit einem generischen Suchmuster kann es zu Konflikten kommen, da beide Teilanfragen prinzipiell widersprüchliche Angaben beinhalten können. Die Konflikte und Möglichkeiten, um diese automatisch zu erkennen und aufzulösen, werden im folgenden Abschnitt untersucht.

7.3.3 Konfliktmanagement

Die Verknüpfung einer ad-hoc-Anfrage mit einem generischen Suchmuster erfolgt im Sinne einer UND-Semantik. Daher können Konflikte auftreten, indem die Anfrage und das generische Suchmuster Angaben enthalten, die sich gegenseitig widersprechen bzw. nicht miteinander kompatibel sind.

Im Folgenden wollen wir nun systematisch die verschiedenen Konfliktfälle aufgreifen und untersuchen, ob und wie diese automatisch aufgelöst werden können. Dabei soll angenommen werden, dass sowohl das gegebene generische Suchmuster als auch die ad-hoc-Anfrage selbst keine Konflikte beinhalten.

Definition 7.4: Die Verknüpfung eines generischen Suchmusters mit einer ad-hoc-Anfrage heißt *konfliktfrei*, wenn keine Konflikte zwischen den Angaben des generischen Suchmusters und den Angaben der ad-hoc-Anfrage auftreten. Dabei wird angenommen, dass sowohl das generische Suchmuster als auch die ad-hoc-Anfrage selbst bereits konfliktfrei sind.

Im Zuge der Klarheit und Nachvollziehbarkeit für den Benutzer ist es anzustreben, dass die auftretenden Konflikte stets in ähnlicher Weise behandelt werden, beispielsweise durch die Priorisierung einer Teilanfrage. Naheliegend ist es beispielsweise, der ad-hoc-Anfrage einen höheren Stellenwert einzuräumen, da diese das aktuelle Informationsbedürfnis beinhaltet und das generische Suchmuster eher eine allgemeine und längerfristige Vorliebe beschreibt.

Im Folgenden werden wir nun die einzelnen MLS-QL-Konstrukte der Reihe nach aufgreifen. Dabei beschreiben wir zunächst, wie die Verknüpfungssemantik bezüglich des jeweiligen Konstrukts definiert werden kann, wie demnach Konflikte aussehen und wie diese behandelt werden können.

In der Regel ist die Verknüpfung an den Stellen, d.h. bei den MLS-QL-Konstukten, unproblematisch, an denen nur eine Teilanfrage Angaben enthält. In diesem Fall werden diese Angaben für die resultierende Rechercheanfrage übernommen. Probleme können dabei lediglich bei der Gruppierung oder der Sortierung auftreten, da diese Konstrukte in Wechselwirkung mit anderen Teilen der Anfrage stehen. Bei den anderen Konstrukten können Konflikte nur auftreten, wenn beide Teilanfragen Angaben gemacht haben.

SELECT: Hier werden zum einen die Attribute aufgezählt, die der Benutzer beim Ergebnis sehen möchte, zum anderen kann die Anzahl der Ergebnistreffer beschränkt werden. Bei der Attributaufzählung entspricht der Verknüpfungssemantik, dass aus den beiden Mengen der gewünschten Attribute die Obermenge gebildet wird. An dieser Stelle können daher keine Konflikte auftreten. Enthält nur eine Teilanfrage einen `TOP(k)`-Operator, wird dieser für die Gesamtanfrage übernommen. Enthalten beide Teile einen `TOP(k)`-Operator mit unterschiedlichem k, wird der Operator von der ad-hoc-Anfrage übernommen.

FROM: In diesem Teil der Anfrage erfolgt die Auswahl der zu befragenden Literaturdienste. Enthalten beide Teile der Anfrage eine Menge von Diensten, wird für die Gesamtanfrage die Obermenge davon gebildet. In diesem Fall können keine Konflikte auftreten.

WHERE: Mit dieser Klausel werden Bedingungen an Attribute formuliert. Die Bedingungen beider Teilanfragen werden mit dem booleschen AND miteinander verknüpft. Widersprüche können entstehen, wenn an das gleiche Attribut verschiedene Bedingungen gestellt werden, die nicht miteinander verträglich sind. Bei numerischen Attributen kann die Verträglichkeit von Bedingungen geprüft werden, indem die Schnittmenge der geforderten Wertebereiche gebildet wird. Ist die Schnittmenge leer, sind die Bedingungen nicht miteinander verträglich und es werden die Bedingungen der ad-hoc-Anfrage übernommen. Bei alphanumerischen Attributen ist dieser Test erheblich schwieriger. Die Bedingungen können mit dem MATCHING-Operator oder einer Menge von Worten, die im Text enthalten oder nicht enthalten sein sollen, beschrieben werden. An dieser Stelle kann nur ein offensichtlicher Widerspruch erkannt werden, nämlich wenn eine Bedingung in positiver und eine andere Bedingung in negativer Form formuliert ist. Ansonsten kann keine automatische Konflikterkennung und damit auch keine automatische Auflösung bzw. Behandlung erfolgen.

DUPLICATES: Zur Bestimmung von Duplikaten werden eine Reihe von Attributen mit zugehörigen Operatoren angegeben. Bei verschiedenen Mengen von Attributen wird die Obermenge der betroffenen Attribute verwendet. Werden auf demselben Attribut unterschiedliche Operatoren verwendet, so dominiert der gewählte Operator in der aktuellen Anfrage.

PREFERRING: An dieser Stelle kann prinzipiell kein Konflikt auftreten, da die Angaben keine Auswirkungen auf das Ergebnis, sondern nur auf den berechneten Relevanzwert der einzelnen Dokumente haben. Selbst widersprüchliche Präferenzen können durch ihre spezielle Gewichtung miteinander kombiniert werden, indem die Relevanzwerte eines Dokuments gemäß beider Präferenzen berechnet und entsprechend aufsummiert werden. In einem Konfliktfall heben sich dann widersprüchliche Präferenzen tendenziell auf.

GROUP BY: Werden in beiden Anfrageteilen Listen von Attributen angegeben, so können Konflikte entstehen. Hier ist es nicht sinnvoll die Obermenge der genannten Attribute zu bilden, da auch die Reihenfolge entscheidend ist. Weiterhin wäre die Bildung einer Schnittmenge zu überlegen. Dabei könnten dann aber auch leicht relativ unwichtige Attribute in Erscheinung treten. Daher werden bei einem Konflikt, d.h. wenn nicht genau die gleichen Attribute in der gleichen Reihenfolge genannt werden, die Attribute der ad-hoc-Anfrage übernommen.

Bei MLS-QL muss das Ergebnis, wenn es eine Gruppierung enthält, auch nach diesen Gruppierungsattributen (oder einer Untermenge davon) sortiert werden. Maximal ein zusätzliches Attribut darf bei der Sortierung angegeben werden. Daher wird im Konfliktfall

auch die Sortierung der ad-hoc-Anfrage mit übernommen. Diese ist ja laut Annahme in jedem Fall verträglich mit der Gruppierungsklausel.

An dieser Stelle ist zusätzlich der Fall zu betrachten, wenn nur eine Teilanfrage eine Gruppierung enthält. Hier wird wie im Konfliktfall gehandelt. Diese Gruppierung wird für die Gesamtanfrage übernommen, zusammen mit der entsprechenden Sortierungsklausel.

ORDER BY: In diesem Fall verhält es sich ähnlich wie bei der Gruppierung. Falls hier zwei unterschiedliche Listen von Attributen vorliegen, können diese schlecht miteinander kombiniert werden. In diesem Fall wird die Sortierung der ad-hoc-Anfrage vorgezogen, sofern bei keiner der Teilanfragen eine Gruppierung vorliegt. Liegt eine Gruppierung vor, so wurde das Vorgehen bzgl. der Sortierung bereits im vorangegangenen Punkt geklärt.

Tabelle 7.1 fasst noch einmal die möglichen Konflikte und Konfliktbehandlungen unter jedem MLS-QL-Konstrukt zusammen. In den Fällen, in denen keine konfliktfreie Verknüpfungsmöglichkeit besteht, wurden im Hinblick auf Klarheit und Nachvollziehbarkeit einheitlich die Angaben der ad-hoc-Anfrage als dominant festgelegt.

	Mögliche Konflikte und Konfliktbehandlungen
`SELECT`	Attributaufzählung: Obermenge `TOP(k1)-TOP(k2)`-Konflikt: ad-hoc-Anfrage dominiert
`FROM`	Quellenaufzählung: Obermenge
`WHERE`	Bedingungen: `AND`-Verknüpfung bei erkanntem Widerspruch: ad-hoc-Anfrage dominiert
`DUPLICATES`	Attributaufzählung: Obermenge bei widersprüchlichen Operatoren: ad-hoc-Anfrage dominiert
`PREFERRING`	Präferenzen: gewichtete `AND`-Verknüpfung
`GROUP BY`	Konflikt: ad-hoc-Anfrage dominiert (samt Sortierungsklausel) Einseitig: entsprechende Teilanfrage dominiert (samt Sortierungsklausel)
`ORDER BY`	Konflikt: ad-hoc-Anfrage dominiert bzw. siehe `GROUP BY`

Tabelle 7.1: Mögliche Konfliktfälle und deren Behandlung

7.4 Gestaltung einer Rechercheschnittstelle

Die Herausforderung bei der Gestaltung einer Anfrageschnittstelle liegt größtenteils in der geeigneten Reduktion der von MLS-QL angebotenen Anfragekomplexität, wobei dem Konzept der generischen Suchmuster eine große Bedeutung zukommt. Die Anfrageschnittstelle soll für den Benutzer einfach und intuitiv zu bedienen sein und gleichzeitig hinreichend viele Ausdrucksmittel für die Formulierung von aussagekräftigen Anfragen zur Verfügung stellen. Die Ergebnispräsentation soll dem Benutzer auch bei großen Ergebnismengen erlauben, einen schnellen Überblick über die vorhandenen Dokumente zu gewinnen. Weiterhin soll sie den Benutzer nicht lange auf die Ergebnisse warten lassen und ihn bei der Entscheidungsfindung für ein Dokument oder für mehrere Dokumente in geeigneter Weise unterstützen.

7.4.1 Gestaltungsgrundsätze

Die Gestaltung einer Rechercheschnittstelle soll sich an folgenden Grundsätzen orientieren:

- Benutzung von WIMP-Gestaltungsmitteln: Die in Abschnitt 3.5.2 vorgestellten Ansätze zur Gestaltung von Rechercheschnittstellen verwenden eine Vielzahl verschiedener graphischer Ausdrucksmittel und Metaphern. Der vorliegende Ansatz soll mit WIMP-Gestaltungsmitteln arbeiten, da das Aussehen und die Funktionalität dieser Komponenten den meisten Benutzern bereits von anderen Anwendungen her bekannt und vertraut sind. Dadurch soll nicht nur eine einfache und intuitive Benutzerinteraktion gewährleistet werden, sondern auch den menschlichen Eigenschaften und Fertigkeiten Rechnung getragen werden: Die Bedienung der WIMP-Komponenten stellt keine hohen Erwartungen an den Benutzer und verzögert die Antwortzeiten des Recherchesystems in der Regel deutlich weniger als ein Einsatz von 3D-Techniken.
- Anwendung von bekannten und bewährten Gestaltungskonzepten: In Abschnitt 3.5.2.3 wurden Ansätze für die Gestaltung von Rechercheschnittstellen vorgestellt, die auch im Rahmen der vorliegenden Arbeit verwendet werden können und sollen, um eine einfache und intuitive Benutzerinteraktion zu gewährleisten. Dabei sollen die Hinweise des Gestaltungsrahmens für Rechercheschnittstellen berücksichtigt werden. Weiterhin sollen die Hi-Cites-Darstellung und interaktive Nachbearbeitungsoperatoren für die Ergebnispräsentation übernommen werden (SenseMaker). Und schließlich soll ggf. der Drag&Drop-Ansatz eingesetzt werden (DLITE).
- Geeignete Beschränkung der MLS-QL-Funktionalität: In den folgenden Abschnitten soll eine Standard-Rechercheschnittstelle für Meta-Recherchesysteme im Bereich der wissenschaftlichen Literaturversorgung entworfen werden. Die Schnittstelle soll dem Benutzer eine sinnvolle Auswahl an Anfragemitteln zur Verfügung stellen, die hinreichend ausdrucksmächtig, aber auch einfach und intuitiv zu bedienen sein sollen.

Benutzerstudien in Abschnitt 3.2.2 haben gezeigt, dass nur ein geringer Prozentsatz der Benutzer als sogenannte „Power-User“ bezeichnet werden können, die sämtliche angebotene Funktionen auch voll ausschöpfen. Für diese Benutzergruppe wäre es sicherlich sinnvoll, zusätzlich eine Expertenschnittstelle anzubieten, über welche die gesamte Funktionalität von MLS-QL zur Verfügung steht. Darauf wird im Rahmen der vorliegenden Arbeit allerdings verzichtet, da der Schwerpunkt und die Herausforderung vielmehr in der Gestaltung einer einfachen und intuitiven Rechercheschnittstelle besteht.

- Kontinuierlicher Ergebnismengenaufbau: Um zumindest die empfundene Wartezeit für den Benutzer zu verkürzen, soll das Konzept des kontinuierlichen Ergebnismengenaufbaus verfolgt werden. Dazu werden die Teilergebnisse, sobald sie von den zugrundeliegenden Diensten empfangen worden sind, sofort an den Benutzer weitergereicht. Das hat zwar Konsequenzen für die Implementierung der Duplikaterkennungs- bzw. Nachbearbeitungsoperatoren, da diese dafür nicht blockierend sein dürfen, ist aber zur Erfüllung der Anforderung „Ungeduld“ für den Benutzer von zentraler Bedeutung.
- Persistenter, persönlicher Arbeitsbereich: Dieser soll zum einen zur Umsetzung der generischen Suchmuster dienen, zum anderen auch Objekte wie Anfragen oder Ergebnisse für den Benutzer sitzungsübergreifend speichern und bei Bedarf wieder zur Verfügung stellen.

Durch die genannten Gestaltungsgrundsätze können bereits zahlreiche Anforderungen erfüllt werden. Tabelle 7.2 gibt einen Überblick über die Anforderungen und die zur Erfüllung vorgesehenen Gestaltungskonzepte.

Anforderung	Konzepte des Lösungsansatzes
Einfache Benutzerinteraktion (A2)	Konzept der generischen Suchmuster Beschränkung der MLS-QL-Funktionalität
Intuitive Benutzerinteraktion (A2)	WIMP-Gestaltungsmittel Einsatz bekannter / bewährter Gestaltungskonzepte
Erinnerungshilfen (A4)	Persistenter, persönlicher Arbeitsbereich Konzept der generischen Suchmuster
Niedrige Erwartungen (A4)	Beschränkung der MLS-QL-Funktionalität WIMP-Gestaltungsmittel Konzept der generischen Suchmuster
Ungeduld (A4)	Kontinuierlicher Ergebnismengenaufbau

Tabelle 7.2: Erfüllung der Anforderungen durch den Lösungsansatz

Die zusätzlichen Anforderungen des Gestaltungsrahmens sind von konkreter Natur, d.h. sie können nicht prinzipiell durch den Einsatz von Konzepten erfüllt werden, sondern sind bei der Realisierung im einzelnen aufzugreifen und umzusetzen.

In den folgenden Abschnitten wird die Umsetzung einer geeigneten Rechercheschnittstelle entwickelt. Zunächst wird eine Anfrageschnittstelle entworfen (7.4.2), dann wird das Konzept der generischen Suchmuster im Rahmen eines persönlichen, persistenten Arbeitsbereichs umgesetzt (7.4.3). Schließlich wird eine geeignete Darstellung für die Ergebnispräsentation entwickelt (7.4.4). Die Entwicklung einer geeigneten Rechercheschnittstelle soll als Realisierungsvorschlag verstanden werden, der seinen Schwerpunkt in erster Linie auf eine geeignete Recherchefunktionalität und weniger auf ausgereifte Designkonzepte legt.

7.4.2 Gestaltung einer Anfrageschnittstelle

Bei der Gestaltung einer geeigneten Anfrageschnittstelle sollen die Hinweise des Gestaltungsrahmens aufgegriffen werden. Dieser regt an, das Anfragefenster zunächst in verschiedene Bereiche zu unterteilen, wobei in jedem Bereich ein spezieller Teil der Anfragefunktionalität angesiedelt werden soll. Der Gestaltungsrahmen schlägt folgende vier Bereiche vor: Literaturquellen, Bedingungen, Ergebnisse und Aktionen. Um die Funktionalität von MLS-QL geeignet auf die verschiedenen Bereiche aufteilen zu können, benötigen wir zusätzlich noch einen weiteren Teil für die Angabe von Präferenzen. Da sich die Präferenzen in Bezug auf die Semantik deutlich von den Anfragebedingungen unterscheiden, sollen sie auch in unterschiedlichen Bereichen positioniert werden. Die Duplikaterkennungs-, Gruppierungs- und Sortierfunktionen können dem Bereich der Ergebnisdarstellung ebenso zugeordnet werden wie die Attributauswahl und die Zahl der gewünschten Ergebnisse. Tabelle 7.3 stellt die Strukturierung der Anfrageschnittstelle und die Zuordnung der einzelnen MLS-QL-Konstrukte zu den unterschiedlichen Bereichen dar.

Bereiche der Anfrageschnittstelle	MLS-QL-Konstrukt
Literaturquellen	`FROM`
Bedingungen	`WHERE`
Präferenzen	`PREFERRING`

Ergebnisse	`SELECT TOP(k)`
	`DUPLICATES`
	`GROUP BY`
	`ORDER BY`

Aktion	–

Tabelle 7.3: Umsetzung der MLS-QL-Funktionalität in Bereichen

Literaturquellen. Dieser Bereich soll dem Benutzer die verfügbaren Informationsquellen verdeutlichen. Dabei wollen wir nicht nur die Namen der Dienste aufführen, sondern durch eine geschickte Anordnung auch auf die verschiedenen Arten von Informationsdiensten aufmerksam machen, die im vorhandenen Meta-Recherchesystem eingebunden sind. Im oberen Teil von Abbildung 7.1 ist die Umsetzung dieses Bereichs zu sehen. Für jede Dienstkategorie gibt es einen separaten Auswahlschalter, mit dem sämtliche Informationsquellen der Klasse aktiviert werden können. Im vorliegenden Fall möchte der Benutzer alle eingebundenen Buchhändler befragen sowie zwei Karlsruher Bibliotheken.

Weiterhin wird die Möglichkeit angeboten, die für die Anfrage geeigneten Informationsquellen automatisch durch einen Trader bestimmen zu lassen. Wird diese Option aktiviert, ist keine explizite Quellenauswahl mehr möglich.

Suchkriterien. In diesem Bereich werden die (harten) Anfragebedingungen angegeben. Die Bedingungen beziehen sich jeweils auf Attribute des Domänenmodells. Die zur Verfügung stehenden Attribute sowie die booleschen Verknüpfungen sollen dem Benutzer explizit präsentiert werden. An dieser Stelle besteht die Möglichkeit, dem Benutzer nur eine Untermenge der Attribute des Domänenmodells anzubieten, falls die Anfragebearbeitung für einige Bedingungen nicht möglich ist oder zu aufwendig sein sollte.

Der zweite Bereich von Abbildung 7.1 zeigt die Realisierung der Anfragebedingungen. Der Benutzer ist an Büchern interessiert, die das Wort „Java“ im Titel enthalten. Zur intuitiveren Benutzbarkeit wurden einige Attribute des Domänenmodells zusammengefasst, z.B. Autor (Vorname, Nachname) oder Verlag (Name, Ort). Für das jeweils ausgewählte Attribut werden in einem separaten Feld Beispiele für Anfrageformulierungen angegeben. Im vorliegenden Bereich werden dem Benutzer zunächst drei Platzhalter für die Formulierung von Bedingungen angeboten. Dabei weist jede Attributauswahl zu Beginn einen anderen Eintrag auf, um dem Benutzer die vielfältigen Möglichkeiten anzudeuten. Auch wenn Benutzerstudien gezeigt haben, dass 80 bis 90 % der Anfragen drei oder weniger Stichworte enthalten (vgl. Tabelle 3.2), besteht über die Schaltfläche mit der Bezeichnung „mehr“ die Möglichkeit, weitere Platzhalter für Anfragebedingungen anzufordern.

Präferenzen. In diesem Bereich werden die Präferenzen angegeben. Diese Funktionalität ist für die Benutzer in der Regel neu und unbekannt. Daher verfolgt die Umsetzung in Abbildung 7.1 einen Ansatz, bei welchem der Benutzer keine selbständigen Eingaben zu machen braucht. Auch in diesem Bereich werden wieder drei Platzhalter für Präferenzen angeboten, wobei die Auswahl der möglichen Attribute hier deutlich eingeschränkter ist als bei der Anfrageformulierung. Es werden nur Attribute angeboten, auf welchen die Spezifikation von Präferenzen auch Sinn macht. Zu jedem Attribut wird eine Auswahl an vorgefertigten Präferenzen präsentiert, wobei die Schwierigkeit sicherlich darin besteht, eine aussagekräftige natürlichsprachliche Formulierung zu finden, die die Wirkweise der Präferenz möglichst gut widerspiegelt. Als Jahr-Präferenzen werden beispielsweise „so aktuell wie möglich“, „so alt

wie möglich“, „in den letzten 5 Jahren“ und „in den letzten 10 Jahren“ angeboten. Als Lieferzeit-Präferenzen stehen „möglichst sofort“, „noch heute“, „in den nächsten 3 Tagen“ und „in den nächsten 1 - 2 Wochen“ zur Verfügung.

Zu jeder Präferenz kann sich der Benutzer über den Detail-Knopf in einem separaten Fenster eine Erklärung des Präferenzkonzepts und die genaue Spezifikation der ausgewählten Präferenz anschauen. In diesem separaten Fenster wird die gewählte Gewichtung der Wertebereiche anzeigt. An dieser Stelle hat der Benutzer die Möglichkeit, die voreingestellte Spezifikation der Präferenz zu verändern. Dazu muss er die Präferenz allerdings neu benennen. Sobald das Fenster wieder geschlossen wird, ist die Präferenz unter dem neuen Namen nun auch in der Präferenzenauswahl sichtbar und einsetzbar, allerdings nur im Rahmen der aktuellen Sitzung. Auf die Möglichkeit einer persistenten Speicherung wird im nächsten Abschnitt eingegangen.

Präferenzen werden über die Vergabe von Gewichten verknüpft. Für die Einstellung der Gewichte wird jeweils ein Schieberegler vorgesehen, mit einer Skala von 0 bis 10. Die Standardeinstellung betrachtet alle angegebenen Präferenzen als gleich wichtig. Der Benutzer kann diese Gewichtung selbständig verändern. Die Gewichte sind im Hinblick auf die MLS-QL-Funktionalität prinzipiell beliebig wählbar. Eine Skalierung von 0 bis 10 wurde gewählt, weil der Benutzer damit sicherlich drei Präferenzen ggf. unterschiedlich gewichten kann. Denkbar wären auch natürlichsprachliche Bezeichnungen wie „sehr wichtig“ und „weniger wichtig“, darauf wurde aus Platzmangel jedoch verzichtet. Prinzipiell wäre es ebenso denkbar, auf die explizite Vergabe von Gewichten auf der Anfrageoberfläche zugunsten der Klarheit und Übersichtlichkeit zu verzichten und für alle Präferenzen dieselben Gewichte festzulegen. Diese Gewichte könnten dann bei Bedarf in einem separaten Detail-Fenster eingesehen und verändert werden. Die Umsetzung mit der sichtbaren Gewichtung wurde in diesem Fall gewählt, um auf die unterschiedliche Konzeption und die weiche Semantik der Präferenzen im Vergleich zu den Suchkriterien aufmerksam zu machen.

Ergebnis. Dieser Bereich der Anfrageschnittstelle soll die Steuerparameter einer Anfrage und genauere Angaben zur Darstellung der Ergebnismenge beinhalten. Das betrifft mehrere Teilkonstrukte von MLS-QL (vgl. Tabelle 7.1), die jeweils auch eine sehr mächtige Funktionalität unterstützen. Daher wird bei der Umsetzung dieses Bereichs nur ein Teil der vorhandenen Funktionen berücksichtigt. Dem Benutzer werden nicht alle möglichen Optionen angeboten, sondern es wird versucht, eine geeignete Auswahl und sinnvolle Voreinstellungen zu finden. Den Voreinstellungen kommt eine besondere Bedeutung zu, da Benutzerstudien gezeigt haben (vgl. Abschnitt 3.2.2.2), dass zwei Drittel der Benutzer die angegebenen Einstellungen übernehmen, unabhängig davon, wie diese Einstellungen gewählt werden.

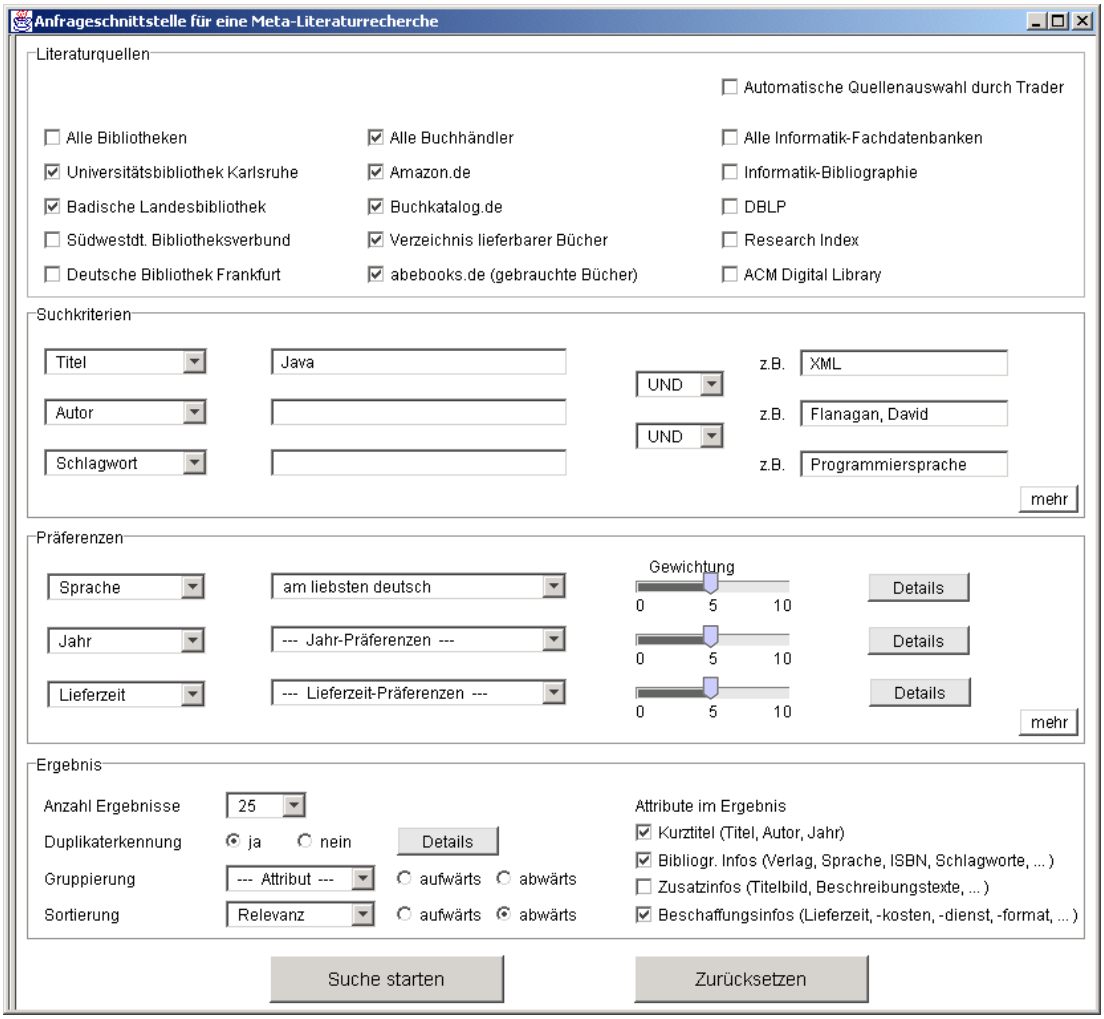


Abbildung 7.1: Realisierung einer Anfrageschnittstelle

Abbildung 7.1 zeigt im unteren Bereich die Umsetzung dieses Funktionsbereichs. Bei der Anzahl der Ergebnisse kann der Benutzer zwischen 10, 25, 50, 100 und sämtlichen gefundenen Ergebnissen wählen. Die angezeigten Attribute im Ergebnis werden nicht einzeln, sondern gruppenweise ausgewählt. Die Gruppen repräsentieren die drei Bereiche des Domänenmodells, bieten aber zusätzlich noch die Kurztitel-Gruppe an, die zu kurzgefassten, dafür aber in der Regel auch schnell erscheinenden Ergebnissen führt.

Bei der Duplikaterkennung kann der Benutzer lediglich entscheiden, ob er eine Duplikaterkennung wünscht oder nicht. Bei der Umsetzung an der Benutzungsschnittstelle wird von der Art und Weise der Duplikaterkennung abstrahiert. Über den Detail-Knopf kann auch hier der Benutzer in einem separaten Fenster die verwendeten Einstellungen ansehen und ggf. verändern. Zur Spezifikation der Duplikaterkennung stehen in dem separaten Fenster nur Attribute zur Verfügung, die in diesem Zusammenhang Sinn machen (d.h. Titel, Autor,

Jahr, Verlag, Sprache, Auflage, Ausgabe und ISBN). Für numerische Attribute können nur exakte Vergleiche gewählt werden, für alphanumerische Attribute steht jeweils ein exakter und ein toleranter Vergleichsoperator zur Verfügung. Zwischen verschiedenen toleranten Vergleichsoperatoren für alphanumerische Attribute wird an der Benutzungsschnittstelle nicht unterschieden. Die Standardeinstellung für die Duplikaterkennung hängt von der Anfrage des Benutzers bzw. von den Attributen ab, die der Benutzer im Ergebnis sehen möchte. Wenn nur Kurztitel angefordert werden, kann zur Duplikaterkennung beispielsweise kein Vergleich der ISBN-Angaben stattfinden.

Bei der Umsetzung der Gruppierung wurde im Hinblick auf Intuitivität und Klarheit die Funktionalität so beschränkt, dass nur ein Attribut ausgewählt werden kann und dazu eine Sortierreihenfolge festgelegt werden sollte. Auch bei der Gruppierung werden nur die Attribute angeboten, die in diesem Zusammenhang Sinn machen. Die Standardeinstellung sieht keine Gruppierung vor. Analog wird bei der Umsetzung der Sortierung ebenfalls nur ein Attribut angeboten. Hier stehen sämtliche Attribute des Domänenmodells zur Verfügung. Zusätzlich wird die Sortierung nach dem Relevanzwert angeboten, welche in dem Moment zur Standardeinstellung wird, wenn mindestens eine Präferenz angegeben wurde.

Abgeschlossen wird die Umsetzung der Anfragefunktionalität mit zwei Schaltflächen, zum Starten einer Suche und zum Zurücksetzen der bisherigen Eingaben.

Bei der Gestaltung der Anfrageschnittstelle wurde in erster Linie auf eine geeignete Funktionalität geachtet, weniger auf Designfragen. Sicherlich lässt sich die entwickelte Anfrageschnittstelle durch entsprechende Farb- und Designparameter noch ansprechender gestalten.

Eine Beschränkung der angebotenen MLS-QL-Funktionalität ist sinnvoll, um dem Benutzer eine einfache und übersichtliche Anfrageschnittstelle anzubieten. Dabei sind bei der Beschränkung die Bedürfnisse des Benutzers zu berücksichtigen sowie die technische Realisierung und die Leistungsparameter des zugrundeliegenden Meta-Recherchesystems. In diesem Abschnitt haben wir die Überlegungen zur Funktionalitätsbeschränkung lediglich auf Basis der Benutzerbedürfnisse angestellt, beispielsweise bei der Auswahl der präsentierten Attribute. Das betrifft die Attribute, für die Bedingungen bzw. Präferenzen angegeben werden können, oder die Attribute, auf deren Basis eine Duplikaterkennung durchgeführt werden kann. Das betrifft ebenfalls die angebotenen Duplikaterkennungsalgorithmen oder die gewählten Standardeinstellungen. An diesen Stellen sind zusätzliche Untersuchungen notwendig, die die technische Realisierung und Performanz des Meta-Recherchesystems berücksichtigen. Daher ist die in diesem Abschnitt getroffene Auswahl der Beschränkungen nur als vorläufig zu sehen, die Beschränkungen sollen nach den in den folgenden Kapiteln angestellten technischen Betrachtungen und den anschließenden Experimenten nochmals aufgegriffen und ggf. angepasst werden.

7.4.3 Gestaltung eines persistenten, persönlichen Arbeitsbereichs

In diesem Abschnitt wird eine Realisierung eines persistenten, persönlichen Arbeitsbereichs entwickelt. Jeder Benutzer soll dadurch die Möglichkeit bekommen, individuelle Einstellungen oder Informationen sitzungsübergreifend zu verwalten und damit stets verfügbar zu haben. Dadurch sollen dem Benutzer in erster Linie Erinnerungshilfen gegeben werden, dem Benutzer soll aber auch eine einfachere Interaktion ermöglicht werden.

Zunächst muss geklärt werden, welche Arten von Einstellungen oder Informationen dauerhaft abgespeichert werden sollen. In Abschnitt 3.3.2 wurde das Personalisierungskonzept von Daffodil vorgestellt, welches nicht nur das Speichern von Anfragen, sondern auch von Ergebnissen erlaubt. Beides soll auch vom vorliegenden Lösungsansatz unterstützt werden. Darüber hinaus wurde im vorangegangenen Abschnitt bereits deutlich, dass ein Benutzer individuelle Präferenzen spezifizieren kann. Diese sollten ihm daraufhin ebenfalls sitzungsübergreifend zur Verfügung stehen. Und schließlich soll dem Benutzer auch sein eigenes Repertoire an generischen Suchmustern zur Verfügung stehen, das er ggf. zur Anreicherung einer ad-hoc-Anfrage verwenden kann. Allerdings soll es nur möglich sein, ein einziges generisches Suchmuster mit einer ad-hoc-Anfrage zu verknüpfen.

Damit sind die vier wesentlichen Konzepte benannt, die zur persistenten Speicherung vorgesehen werden sollen: Anfragen, Ergebnisse, Präferenzen und generische Suchmuster. Zu jedem Konzept können eine Reihe von Objekten gespeichert werden, beispielsweise eine Reihe von Anfragen oder eben eine Menge von generischen Suchmustern. Dabei kann es durchaus hilfreich sein, noch weitere Strukturierungsmittel innerhalb der Objekte eines Konzepts vorzusehen, z.B. eine hierarchische Anordnung in Ordnern und Unterordnern, in denen jeweils ähnliche Objekte zusammengefasst werden können. Daher wird als Realisierungsform für den persistenten, persönlichen Arbeitsbereich eine Baumstruktur vorgeschlagen. Diese soll dem Benutzer permanent neben der üblichen Rechercheschnittstelle zur Verfügung stehen.

Abbildung 7.2 zeigt die bereits bekannte Anfrageschnittstelle mit dem nun ergänzten persistenten, persönlichen Arbeitsbereich. Die Ordner der obersten Ebene geben die unterschiedlichen Konzepte vor, die der Benutzer dann mit seinen persönlichen Objekten befüllen kann. Im Folgenden wird zu jedem Konzept die Interaktion mit der Anfrageschnittstelle erläutert.

Anfragen. Wählt der Benutzer eine abgespeicherte Anfrage aus, so werden die entsprechenden Angaben und Einstellungen in den dafür vorgesehenen Feldern der Anfrageschnittstelle dargestellt. Sind vorher andere aktuelle Eingaben in den Feldern gemacht worden, so werden diese dadurch überschrieben. Der Aktionsleiste wurde eine zusätzliche Schaltfläche zum Speichern einer Anfrage hinzugefügt. Eine Anfrage kann nur abgespeichert werden, wenn sie mindestens eine Quellenangabe und eine Bedingung (an Titel, Autor,

Schlagwort oder ISBN) enthält. Bei den Attributen im Ergebnis sieht die Realisierung vor, dass immer mindestens eine Gruppe markiert sein muss.

Ergebnisse. Unter dem Konzept der Ergebnisse werden keine einzelnen gefundenen Treffer gesammelt, sondern ganze Ergebnismengen. Wählt der Benutzer eine Ergebnismenge aus, so wird diese in der Ergebnisdarstellung angezeigt. Die Ergebnispräsentation wird im folgenden Abschnitt entwickelt und vorgestellt. Für detailliertere Informationen sei also auf den nächsten Abschnitt verwiesen.

Präferenzen. Im vorangegangenen Abschnitt wurde bereits erwähnt, dass der Benutzer die vorgegebenen Präferenzen auch individuell verändern kann. Durch den persistenten Bereich können diese individuellen Einstellungen auch sitzungsübergreifend gespeichert und wiederverwendet werden. Präferenzen beziehen sich immer auf ein bestimmtes Attribut des Domänenmodells. Die persönlichen Präferenzen werden dem Benutzer automatisch in der Auswahlbox zum entsprechenden Attribut zur Verfügung gestellt, so dass der Benutzer ggf. auch auf diese Weise daran erinnert wird, dass er zu einem bestimmten Attribut bereits eine eigene Präferenz angelegt hat. Wählt der Benutzer eine persönliche Präferenz im persönlichen Bereich aus, so erscheint dasselbe separate Fenster mit der Detailansicht der Präferenz wie beim Betätigen des entsprechenden Detail-Knopfes.

Suchstrategien. Auf die Verwendung des etwas abstrakten und für Benutzer sicherlich ungewohnten Begriffs des generisches Suchmusters wurde an der Benutzungsschnittstelle verzichtet. Dafür wurde der gebräuchlichere Begriff der Suchstrategie gewählt.

Der Aktionsleiste wurde eine weitere Schaltfläche zum Speichern einer Suchstrategie hinzugefügt. Dadurch werden sämtliche Einstellungen der Anfrageschnittstelle als Suchstrategie abgespeichert. An dieser Stelle kann kein automatischer Überprüfungsmechanismus angeboten werden (vgl. Abschnitt 7.3.2), so dass der Benutzer selbst für die Sinnhaftigkeit der Einstellungen verantwortlich ist.

Wählt der Benutzer eine Suchstrategie aus, so werden alle gespeicherten Angaben in den Feldern der Anfrageschnittstelle angezeigt. Sind vorher andere aktuelle Eingaben in den Feldern gemacht worden, so werden diese dadurch überschrieben. Soll eine Suchstrategie mit den Einstellungen einer ad-hoc-Anfrage verknüpft werden, so muss der Drag&Drop-Mechanismus dafür angewendet werden. Zur Verdeutlichung des Verknüpfungskonzepts wird für die Einträge und Einstellungen des generischen Suchmusters eine andere Farbe[10] gewählt (blau) als für die Einträge der ad-hoc-Anfrage (schwarz). Auftretende Konfliktfälle werden automatisch wie in Abschnitt 7.3.3 beschrieben behandelt.

In Abbildung 7.2 wird das in Kapitel 6 eingeführte Beispiel 6.1 mit der Verwendung von generischen Suchmustern noch einmal aufgegriffen. Der Benutzer gibt ad-hoc nur das

[10] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

Stichwort Java ein. Ansonsten findet er in seinem persönlichen Repertoire das generische Suchmuster Lehrbücher vor, welches er mit seiner ad-hoc-Anfrage via Drag&Drop verknüpft. Dadurch erscheinen automatisch eine Reihe von zusätzlichen Einträgen, die alle in blauer Farbe dargestellt sind. D.h. das generische Suchmuster gibt in diesem Fall die Quellen vor, zwei Bibliotheken vor Ort und alle verfügbaren Buchhändler, verwendet drei Präferenzen, von denen zwei vom Benutzer selbst definiert sind und eine aus der vorgegebenen Auswahl stammt. Schließlich besagt die Suchstrategie auch, dass eine Duplikateliminierung gewünscht ist und die Sortierung nach der Relevanz der Ergebnisse erfolgen soll.

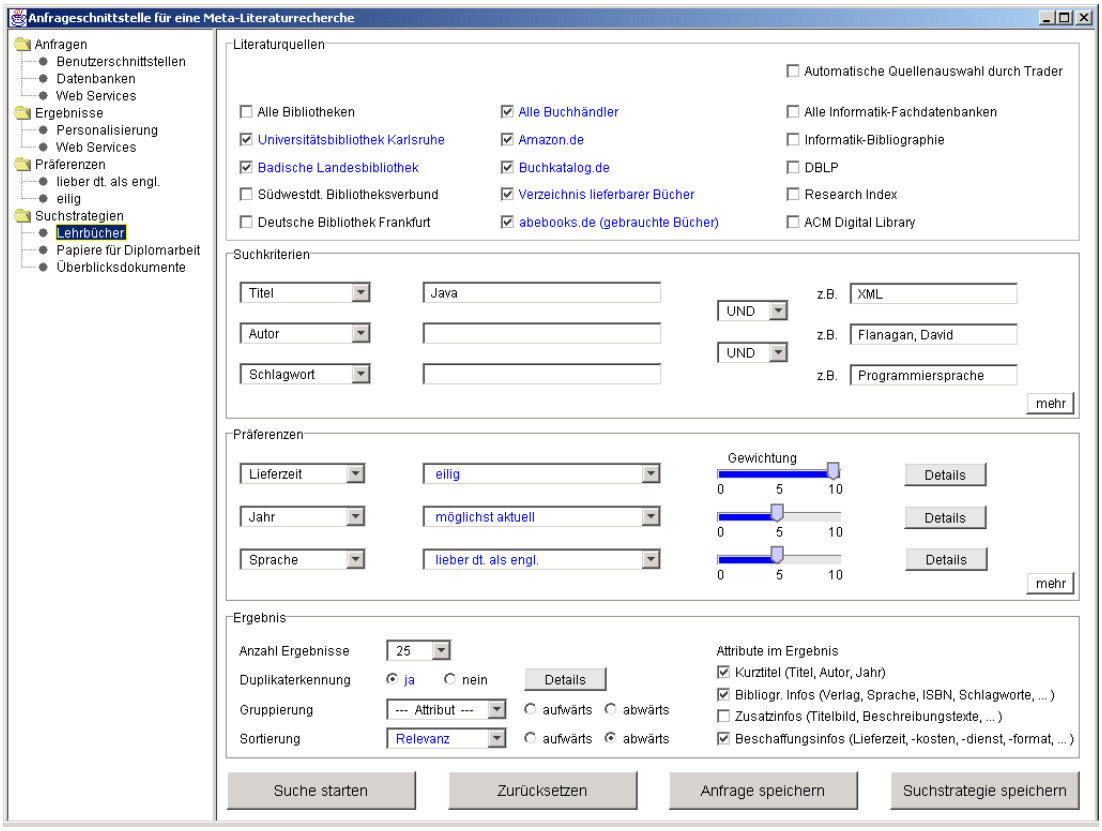


Abbildung 7.2: Realisierung eines persistenten, persönlichen Bereichs

7.4.4 Gestaltung der Ergebnispräsentation

In diesem Abschnitt wird eine geeignete Form der Ergebnisdarstellung entwickelt. Die Ergebnismenge soll in kontinuierlicher Weise aufgebaut werden, um die Wartezeit für den Benutzer zu verkürzen. Dazu ist die Entscheidung zu treffen, in welchen Intervallen die Anzeige der Ergebnismenge aktualisiert werden soll. Wird dieses Intervall zu klein gewählt, wirkt der Ergebnismengenaufbau für den Benutzer leicht sehr unruhig, da die vorläufige Rangierung der Ergebnisse von den neu eintreffenden Ergebnissen stark beeinflusst werden

kann. Wird das Intervall zu groß gewählt, ist der Nutzen des kontinuierlichen Ergebnismengenaufbaus für den Benutzer recht gering. Zur Bestimmung dieses Zeitintervalls können sicherlich ausführlichere Untersuchungen angestellt werden, wir verwenden hier die vorläufige Einstellung eines 10-Sekunden-Takts.

Da die Ergebnismenge gruppiert werden kann, d.h. da Gruppen von Dokumenten gebildet werden können, bietet sich auch für die Ergebnisdarstellung eine Baumstruktur an (siehe Abbildung 7.3). Mit den aktuellen Beschränkungen an der Anfrageschnittstelle kann maximal eine Hierarchiestufe durch die Gruppierung entstehen. Eine weitere Stufe könnte durch die Duplikaterkennung hinzukommen, falls für jede Gruppe von Duplikaten ein Repräsentant erzeugt wird und unter diesem Repräsentanten dann die einzelnen Dokumente mitgeführt werden.

Auf der Ergebnismenge werden die interaktive Gruppierung und Sortierung in derselben Form und mit der gleichen Platzierung im Fenster angeboten wie bei der Anfrageschnittstelle. Die Nachbearbeitungsoperationen können jederzeit ausgelöst werden, indem die gewünschte Einstellung gemacht und anschließend die Schaltfläche zum Aktualisieren der Anzeige betätigt wird.

Die Verwendung der interaktiven Nachbearbeitungsoperatoren gibt Hinweise auf die Attribute, die der Benutzer für seine Entscheidungsfindung heranzieht. Daraus können beispielsweise Vorschläge für die Spezifikation geeigneter generischer Suchmuster abgeleitet und dem Benutzer bei Bedarf angeboten werden.

Die Dokumentreferenzen werden in einheitlicher Form angezeigt. Zunächst wird der Kurztitel angegeben, wobei der Titeleintrag hervorgehoben wird. In der nächsten Zeile werden die bibliographischen Daten angegeben. Die letzte Zeile ist für die Beschaffungsalternativen vorgesehen. Zusätzlich wurde die Hi-Cites-Funktionalität realisiert. Damit können jeweils dieselben Eigenschaften aller Dokumente farblich[11] hervorgehoben werden, wenn der Benutzer einen Augenblick mit dem Mauszeiger über einer Angabe verweilt. In Abbildung 7.3 wird der Benutzer dadurch beispielsweise gerade beim Vergleichen die Lieferzeiten der gefundenen Dokumente unterstützt.

Durch die Schaltfläche zum Speichern der aktuellen Ergebnismenge kann der Benutzer seinen persönlichen Arbeitsbereich erweitern. Wählt der Benutzer darin eine Ergebnismenge an, so wird sie in der beschriebenen Präsentationsform angezeigt. Die Präsentation der Ergebnismenge könnte noch dahingehend erweitert werden, dass der Benutzer nicht nur ganze Ergebnismengen, sondern auch ausgewählte Dokumente von Ergebnismengen abspeichern kann.

Der Benutzer kann jederzeit über die Schaltfläche „Neue Anfrage“ zur Anfrageschnittstelle wechseln.

---

[11] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

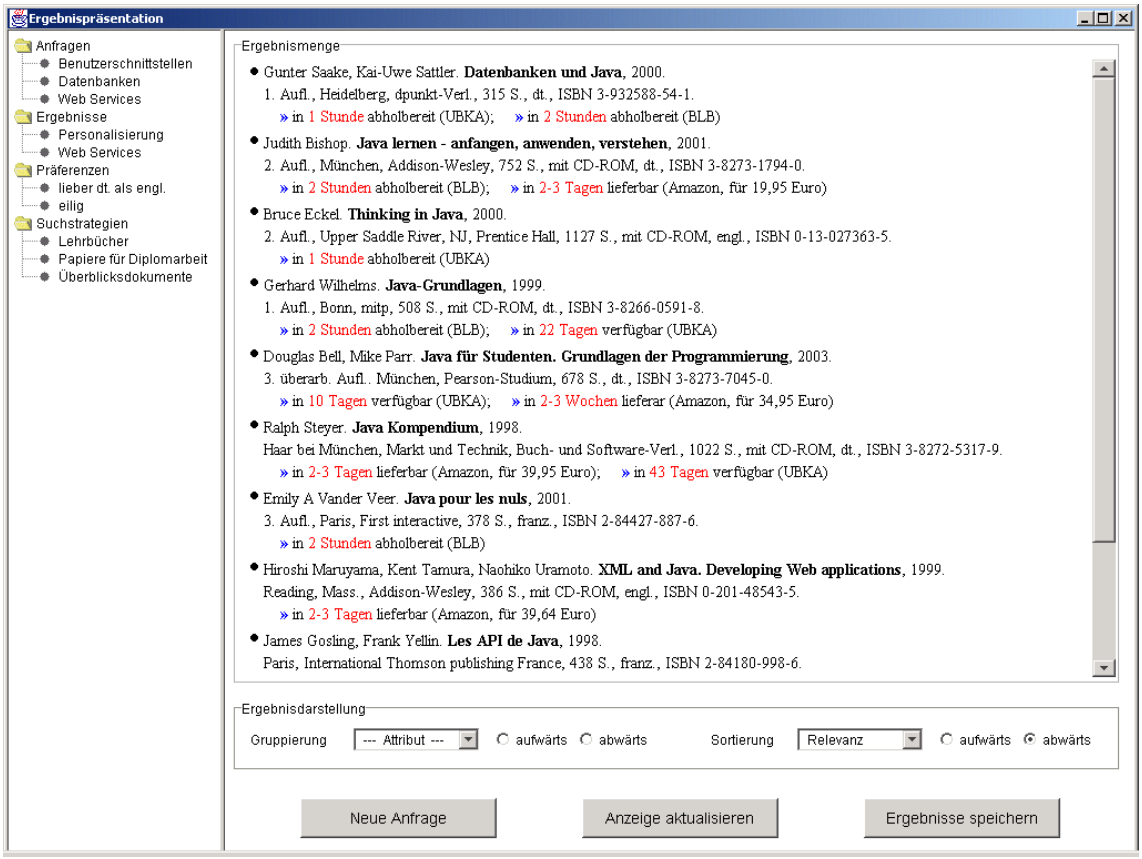


Abbildung 7.3: Realisierung einer Ergebnispräsentation

7.5 Resümee

Dieses Kapitel beschließt den ersten Teil der Arbeit, der sich mit der Konzeption und Gestaltung einer geeigneten Benutzerunterstützung für ein Meta-Recherchesystem im Bereich der wissenschaftlichen Literaturrecherche und -beschaffung beschäftigt.

Dazu wurde in Kapitel 5 ein geeignetes Domänenmodell zur einheitlichen Repräsentation der Dokumente entworfen. Dieses Domänenmodell enthält die Konzepte, die von Benutzern bei der Literaturrecherche und der Entscheidungsfindung benötigt und berücksichtigt werden. In Kapitel 6 wurde eine hinreichend mächtige Anfragefunktionalität erarbeitet und in Form einer Anfragesprache namens MLS-QL spezifiziert. MLS-QL enthält neben den üblichen Anfragekonstrukten auch die Möglichkeit, Präferenzen im Sinne von „lieber-als“-Aussagen anzugeben. Derartige Präferenzen entsprechen der Denkweise des Benutzers und können den Informationswunsch des Benutzers in geeigneter Weise konkretisieren.

Im vorliegenden Kapitel wurde für die mächtige Anfragesprache MLS-QL eine einfache und intuitive Umsetzung an der Benutzungsschnittstelle erarbeitet. Dazu wurde das Konzept der generischen Suchmuster entwickelt, mit welchem die Komplexität der Anfrageformulierung für den Benutzer reduziert werden kann. Anschließend wurde eine Rechercheschnittstelle entworfen, bestehend aus Anfrageschnittstelle, Ergebnispräsentation und einem persistenten, persönlichen Bereich. Nach dieser Kapiteltrilogie können die Anforderungen, die eine geeignete Benutzerunterstützung betreffen (A1 - A4), als erfüllt betrachtet werden.

In den vergangenen Kapiteln wurden bereits einige Fragen in Bezug auf die technische Realisierung und die Performanz des Meta-Recherchesystems aufgeworfen. Diese Themen werden nun in den folgenden Kapiteln des zweiten Schwerpunkts der Arbeit, also bei der Konzeption und Umsetzung einer geeigneten Anfragebearbeitung, vertieft.

8 Eigenschaften der Literaturdienste

8.1 Überblick

Mit diesem Kapitel beginnt nun der zweite Teil der Arbeit, der sich mit der Konzeption und Umsetzung einer geeigneten Anfragebearbeitung beschäftigt. Dazu werden zunächst einige Grundlagen benötigt, nämlich detaillierte Untersuchungen zu den Eigenschaften der existierenden Literaturdienste, die im vorliegenden Kapitel erarbeitet und durchgeführt werden.

Wie in Kapitel 4 bereits motiviert, benötigen die Komponenten der Anfragebearbeitung eine Reihe an Informationen (Metadaten) über die eingebundenen Literaturdienste. Wandler werden im vorliegenden Ansatz lediglich als Transformatoren betrachtet, die die semantische und syntaktische Heterogenität der eingebundenen Dienste überwinden. Die von der Anfragebearbeitung benötigten Informationen betreffen die Eigenschaften der Dienste, die Einflüsse auf die Gesamtantwortzeit des Meta-Recherchesystems haben. Teilweise sind die Eigenschaften der Dienste sicherlich als gegeben hinzunehmen, teilweise können sie aber auch von der Anfragebearbeitung beeinflusst und gesteuert werden.

Im vorliegenden Kapitel werden zunächst die strukturellen Eigenschaften von existierenden Literaturdiensten betrachtet (Abschnitt 8.2). Das sind im Wesentlichen die horizontalen und vertikalen Informationsportionierungen, aber auch die Reihenfolgen, in denen die Literaturdienste ihre Ergebnisse liefern, und die Attribute, die zur Formulierung von Anfragen verwendet werden können.

Anschließend werden Zeitmessungen an existierenden Literaturdiensten durchgeführt, um Aussagen über ihr Antwortzeitverhalten gewinnen zu können, besonders im Hinblick auf die identifizierten Informationsportionen (Abschnitt 8.3). Dieses Antwortzeitverhalten soll formal beschrieben werden (Abschnitt 8.4), um es einerseits in Form von aussagekräftigen Metadaten ablegen und bei der Anfragebearbeitung berücksichtigen zu können und um damit andererseits auch beliebige, aber eben typische Antwortzeitverhältnisse simulieren zu können.

Das Kapitel schließt mit der Formulierung eines Metadatensatzes zur Beschreibung von Literaturdiensten. Dies geschieht durch die Angabe einer entsprechenden DTD (Abschnitt 8.5).

8.2 Strukturelle Eigenschaften der Literaturdienste

Das zentrale Problem im Umgang mit den Diensten im Bereich der Literaturrecherche ist die Informationsportionierung. Dies ist in der Arbeit auch bereits mehrfach angeklungen. Die eingebundenen Literaturdienste werden über ihre Webschnittstelle angesprochen und liefern daher die angeforderten Informationen in mehreren kleinen Portionen zurück.

Jede Informationsportion muss über einen entsprechenden HTTP-Aufruf explizit angefordert werden. Daher verfolgt die vorliegende Arbeit den Ansatz, der Komponente der Anfragebearbeitung alle relevanten Informationen über die Informationsportionen zur Verfügung zu stellen, damit sie auf dieser Basis eine gegebene Anfrage best- und schnellstmöglich planen und ausführen kann.

In diesem Abschnitt untersuchen wir nun die Informationsportionierungen existierender Literaturdienste. Dazu betrachten wir horizontale und vertikale Informationsportionierungen (vgl. Definition 4.1 und 4.2). Zusammen mit der horizontalen Informationsportionierung betrachten wir auch die Sortierreihenfolge, die die Literaturdienste auf den Kurztiteln der Ergebnismenge standardmäßig anbieten bzw. zur Wahl stellen. Im Zuge der vertikalen Informationsportionierung berücksichtigen wir nicht nur die Aufteilung der Attribute auf verschiedene Informationsportionen, sondern identifizieren auch die Attribute, die für die Formulierung von Anfragen zur Verfügung stehen.

8.2.1 Auswahl existierender Literaturdienste

In einem integrierten System zur Literaturrecherche können zahlreiche Informationsdienste von Interesse sein. Wir wollen uns in der vorliegenden Arbeit auf Dienste beschränken, die eine kostenfreie Recherche anbieten. Bei der Zusammenstellung der Literaturdienste sollen sowohl allgemein ausgerichtete Dokumentbestände als auch Dokumentbestände mit einem Schwerpunkt im Fachbereich der Informatik berücksichtigt werden. Von den Diensten soll nicht nur die Recherche, sondern auch die Beschaffung von Dokumenten unterstützt werden. Im Rahmen einer wissenschaftlichen Literaturrecherche sind sowohl Bücher als auch Artikel von Bedeutung. Daher sollen diese Dokumentarten auch im Bestand der Dienste vorhanden sein.

Für eine repräsentative Auswahl werden zehn existierende Literaturdienste betrachtet. Im Bereich der Bücher berücksichtigen wir drei Online-Buchhändler und drei Bibliotheken. Diese Dienste besitzen einen allgemein ausgerichteten Dokumentbestand, enthalten aber auch zahlreiche Bücher im Bereich der Informatik. Darüber hinaus betrachten wir vier Dienste, die sich auf Forschungsartikel (Zeitschriftenaufsätze, Konferenzbeiträge und Technische Berichte) im Bereich der Informatik konzentrieren.

Tabelle 8.1 gibt eine Übersicht über die ausgewählten Literaturdienste. Für jeden Dienst wird ein entsprechendes Dienstkürzel vergeben, mit welchem der Dienst im folgenden Text angesprochen wird. Neben der Webadresse wird von jedem Dienst kurz der Dokumentbestand charakterisiert, im Hinblick auf die fachliche Ausrichtung und den Publikationstyp der vorgehaltenen Dokumente. Dabei gelten folgende Abkürzungen: B (Bücher), Z (Zeitschriften), KB (Konferenzbände), ZA (Zeitschriftenartikel), KA (Konferenzartikel) und TB (Technische Berichte).

Kürzel	Dienstname	URL	Dokumentbestand
AMAZON	Amazon, Deutschland	http://www.amazon.de	(in D lieferbare) B
KNO&KV	Koch, Neff & Ötinger GmbH und Köhler & Volckmar GmbH	http://www.buchkatalog.de	(in D lieferbare) B
LOB	Lehmanns Online Bookshop	http://www.lob.de	(in D lieferbare) B
UBKA	Universitätsbibliothek, Karlsruhe	http://www.ubka.uni-karlsruhe.de	B, KB, Z
BLB	Badische Landesbibliothek, Karlsruhe	http://sua.blb-karlsruhe.de	B, Z
DBF	Deutsche Bibliothek, Frankfurt	http://dbf-opac.ddb.de	B, KB, Z
ACM-DL	ACM Digital Library	http://www.acm.org/dl	KB, KA, Z, ZA von ACM; für Informatik
CSBIB	Informatik-Bibliographie	http://liinwww.ira.uka.de/bibliography	KA, ZA, TB; für Informatik
DBLP	Digital Bibliography & Library Project	http://dblp.uni-trier.de	B, Z, ZA, KB, KA; für Informatik
RI	ResearchIndex	http://citeseer.nj.nec.com/cs	ZA, KA; für Informatik

Tabelle 8.1: Repräsentative Auswahl von Literaturdiensten

8.2.2 Horizontale Informationsportionierung und Ergebnisreihenfolge

Jeder Literaturdienst liefert als Antwort auf eine Anfrage eine Menge von Ergebnistreffern. Diese Menge der Ergebnistreffer wird i.d.R. über mehrere HTML-Seiten verteilt. In diesem Abschnitt soll untersucht werden, wie viele Treffer die ausgewählten Literaturdienste in einer Informationsportion präsentieren. Zusätzlich soll berücksichtigt werden, ob die Anzahl der gleichzeitig angezeigten Treffer variabel ist. Teilweise bieten die Dienste dem Benutzer direkt eine Variationsmöglichkeit an. Teilweise ist es möglich, die angezeigte Trefferanzahl dadurch zu verändern, dass der in der URL mitgegebene Parameter zur Spezifikation der Ergebnisportion entsprechend verändert wird. Diese Vorgehensweise ist zwar nicht von Benutzern, durchaus aber von geeignet definierten Wandlern zu erwarten. Die Berücksichtigung von verschieden großen Ergebnisportionen kann für die Anfragebearbeitung von Bedeutung sein, wenn beispielsweise kleine oder große Informationsportionen im Verhältnis zu ihrer Größe die Ergebnisse wesentlich schneller oder langsamer liefern.

Tabelle 8.2 gibt einen Überblick darüber, wie viele Treffer die ausgewählten Dienste in einer Informationsportion präsentieren (Stand März 2003). Die erste Spalte gibt die Anzahl der Ergebnistreffer an, die standardmäßig in einer Informationsportion geliefert werden. In der Spalte wird ebenso angegeben, in welcher Weise der Benutzer die Trefferanzahl direkt variieren kann. In der nächsten Spalte wird vermerkt, ob die Anzahl der präsentierten Ergebnisse über geschickte Ersetzungen in der entsprechenden URL vergrößert werden kann.

In der letzten Spalte von Tabelle 8.2 wird zunächst die standardmäßige Sortierung der Ergebnistreffer angegeben, anschließend werden die anderen Sortiermöglichkeiten des entsprechenden Dienstes aufgeführt, die dem Benutzer zur Verfügung stehen. Einige Sortierungen lassen sich direkt bei der Begutachtung der Ergebnistreffer nachvollziehen, beispielsweise die alphabetischen Sortierungen nach Titeln oder Autoren sowie die Sortierungen nach dem Erscheinungsjahr. Diese Sortierungen beruhen auf Werten von einzelnen Attributen, die dem Benutzer auch zugänglich sind. Die Wirkweise anderer Sortierungen lässt sich nicht so gut nachvollziehen. Sie basieren auf internen Berechnungen, die offensichtlich mehrere Kriterien berücksichtigen. Die „besten Ergebnisse“ bei AMAZON scheinen sich aus einer Kombination von Verkaufsrang und den Benutzerbewertungen herzuleiten, das Sortierkriterium „Hubs“ bei RI setzt sich vermutlich aus einer Kombination der Anzahl der in einem Dokument enthaltenen Referenzen und enthaltenen Wörtern wie survey, tutorial bzw. introduction zusammen. Bei LOB lässt sich keine definitive Aussage über die verwendete Sortierung treffen, jedenfalls werden aktuelle und schnell verfügbare Bücher bevorzugt. Bei DBLP erfolgt die Ausgabe der Ergebnistreffer aufgrund der intern vergebenen Kürzel der Konferenzen, in denen die jeweiligen Artikel erschienen sind. Bei CSBIB kann nicht einmal eine Vermutung zur Sortierreihenfolge formuliert werden. Genaue Angaben über die verwendeten Sortierungen sind bei den entsprechenden Diensten nicht zu finden.

	Standardeinstellung der Trefferanzahl; Variationen durch Benutzer	Maximale Trefferanzahl (mittels URL-Anpassung)	Standardeinstellung der Ergebnissortierung; andere Sortiermöglichkeiten
AMAZON	10	25	Beste Ergebnisse; Preis (auf/ab), Datum (ab), Benutzerbewertungen (ab), Titel alphabetisch (auf/ab)
KNO&KV	30	–	Autor alphabetisch (auf); Titel/Verlag alphabetisch (auf), Datum (ab)
LOB	20; beliebig	–	?
UBKA	30	70	Jahr (ab)
BLB	30	70	Jahr (ab)
DBF	10	–	Jahr (ab)
ACM-DL	20	–	Ähnlichkeit mit Anfrage; Titel/Zeitschrift/Konferenz alphabetisch (auf), Datum (ab)
CSBIB	40; 10, 100, 170	–	?
DBLP	100; 10, 50, 200	beliebig	(nach intern vergebenem Konferenzkürzel)
RI	20	50	Anzahl erwarteter Zitierungen (ab); Anzahl absoluter Zitierungen (ab), Hubs (ab), Nutzung (ab), Datum der Indizierung (ab)

Tabelle 8.2: Übersicht über horizontale Informationsportionen und Ergebnisreihenfolgen

8.2.3 Vertikale Informationsportionierung und anfragbare Attribute

Jeder Literaturdienst liefert eine Reihe von Informationen bzw. Attributen zu einem Dokument. In diesem Abschnitt soll die portionsweise Lieferung der verfügbaren Attribute untersucht werden (vertikale Informationsportionierung). Dazu berücksichtigen wir die Attribute, die vom Meta-Recherchesystem angefordert werden, also die Attribute des Domänenmodells.

In Tabelle 8.3 ist eine Übersicht darüber zu finden, welche Attribute bei welchem Dienst in welcher Informationsportion zu finden sind (Stand März 2003). Die eingetragene Zahl gibt an, wie viele HTTP-Aufrufe (mindestens) zum Erhalt der Attribute notwendig sind. Die Kurztitelliste enthält neben einigen Attributen zu den Dokumenten auch jeweils eine URL, die angegeben werden muss, um weitere Attribute zu einem Dokument zu bekommen. Auf dieser HTML-Seite befinden sich dann möglicherweise weitere URLs, die zum Anfordern von zusätzlichen Attributen benutzt werden können. Bei der vertikalen Informationsportionierung liegt also eine gewisse Abhängigkeit zwischen den Informationsportionen vor, d.h. gewisse Informationsportionen können erst angefordert werden, wenn zuvor andere Informationsportionen bereits empfangen wurden. Daher wollen wir im Folgenden nicht nur von Informationsportionen, sondern auch von Informationsebenen bzw. -stufen sprechen. Auf eine Anfrage hin werden zuerst die Informationen der Ebene 1 (Kurztitelliste) präsentiert. Daraus können dann Informationen der Ebene 2 gewonnen werden, daraus dann wiederum Informationen der Ebene 3, usw. Die Stufe der Ebene entspricht damit auch der Anzahl der Verweise, die ein Benutzer an der Webschnittstelle des Dienstes verfolgen muss, um die entsprechenden Informationen bzw. Attribute zu beziehen.

Die ausgewählten Dienste beinhalten in ihrem Bestand verschiedene Publikationstypen. Unterschiedliche Publikationstypen weisen prinzipiell unterschiedliche Attribute zur Beschreibung auf (vgl. Tabelle 5.3). Für die Erstellung der Übersicht betrachten wir für Buchhändler und Bibliotheken die verfügbaren Attribute eines Buches, für die wissenschaftlich ausgerichteten Artikeldienste betrachten wir die verfügbaren Attribute von Konferenz- und Zeitschriftenartikeln. Die prinzipiell nicht verfügbaren Attribute sind in der Tabelle durch „–“ markiert.

Tabelle 8.3 geht davon aus, dass die Informationsportionierung anhand eines Dokuments betrachtet wird, welches mit allen prinzipiell verfügbaren Attributen beschrieben wird. Natürlich müssen Dokumentreferenzen nicht immer alle Attribute aufweisen. Wenn die Attributangaben allerdings vorhanden sind, dann erscheinen sie auf der in der Tabelle angegebenen Ebene.

Bei der Betrachtung von Tabelle 8.3 werden die Unterschiede zwischen den Arten der Dienste sichtbar. Buchhändler liefern mehr Attribute zu den einzelnen Dokumenten als Bibliotheken, besonders im Bereich der inhaltlichen Zusatzinformationen. Zudem präsentieren sie mehr Attribute auf der ersten Ebene, gerade auch die Beschaffungsinformationen. Die Unterschiede zwischen den einzelnen Buchhändlern liegen vor allem in der Menge der verfügbaren

Zusatzinformationen, wobei AMAZON dabei führend ist. AMAZON bietet zu den meisten Dokumenten Beschreibungen und Beurteilungen an, wohingegen bei KNO&KV nur zu wenigen Dokumenten Beurteilungen zu finden sind. Das Inhaltsverzeichnis und die Beschreibungstexte beginnen bei AMAZON auf Ebene 2. Wenn sie vollständig berücksichtigt werden sollen, muss i.d.R. noch Ebene 3 bezogen werden. Die zahlenmäßige Benutzerbeurteilung ist bei AMAZON bereits auf der ersten Ebene zu finden, die Leserrezensionen beginnen dann auf Ebene 2 und können sich auch hier wieder über eine dritte Ebene erstrecken, wenn sie einen gewissen Umfang überschreiten.

Bei Bibliotheken sind hauptsächlich bibliographische Informationen zu den Dokumenten zu finden, wobei der Ladenpreis des Buches nur bei DBF mit angegeben wird. Die Beschaffungsinformationen zu einem Dokument sind i.d.R. nur registrierten Benutzern zugänglich und müssen über mehrere Stufen abgerufen werden. Ebene 3 ist bei UBKA und DBF für die Eingabe der Benutzernummer und des Passworts vorgesehen. Da der UBKA-Dienst sitzungsbasiert ist, muss im Rahmen einer Recherche nur einmal die Benutzernummer und das Passwort angegeben werden. In den folgenden Anfragen erscheinen die Beschaffungsinformationen dann sofort auf Ebene 3. Falls es beim UBKA-Dienst verschiedene Beschaffungsoptionen zu einem Dokument gibt, beispielsweise das sofortige Einsehen im Lesesaal oder das Vormerken eines verliehenen Exemplars im Ausleihbetrieb, kommt eine zusätzliche Ebene hinzu, auf der die unterschiedlichen Signaturen des betreffenden Dokuments präsentiert werden. Diese Signaturen deuten bereits die unterschiedlichen Beschaffungsmöglichkeiten an (LS=Lesesaal, FH=Freihand, MA=Magazin). Trotzdem muss für exakte Informationen (z.B. Lesesaal 2.OG oder verfügbar in 6 Wochen) eine weitere Informationsportion beschafft werden (im schlimmsten Fall also im 5. Schritt). Der BLB-Dienst zeigt auf Ebene 3 einen Teil der Beschaffungsinformationen an, nämlich ob ein Dokument sofort verfügbar ist oder nur vorgemerkt werden kann. Um die genaue Wartezeit für eine Vormerkung zu erfahren, muss Ebene 4 konsultiert werden.

Bei ACM-DL und CSBIB besteht die Möglichkeit, Einfluss auf die Portionierung zu nehmen. Bei Bedarf können mehr Attribute auf Ebene 1 präsentiert werden. Wird bei CSBIB diese Option gewählt, erscheinen alle Attribute zu einem Dokument bereits auf der ersten Ebene. Ansonsten werden nur Titel, Autor und Jahr auf der ersten Ebene und die anderen Attribute auf der zweiten Ebene präsentiert. Bei ACM-DL bezieht die ausführlichere Ergebnis-darstellung lediglich die Kurzfassungen der Dokumente auf der ersten Ebene mit ein. Im Vergleich zu Büchern werden bei Artikeln nur selten Zusatzinformationen angegeben. Das liegt vermutlich daran, dass der Volltext oft gar nicht so lang und meist auch direkt verfügbar ist. ACM-DL sieht zwar prinzipiell Beurteilungen (reviews) von Artikeln vor, diese sind aber bislang nur in den seltensten Fällen verfügbar. Dafür werden für Artikel die Zitierungsinformationen besonders berücksichtigt und ausgewiesen. Es wird sowohl berücksichtigt, welche Dokumente in einem gegebenen Artikel zitiert werden, als auch in welchen Dokumenten der gegebene Artikel zitiert wird.

	AMA-ZON	KNO &KV	LOB	UBKA	BLB	DBF	ACM -DL	CS BIB	DBLP	RI
Titel	1	1	1	1	1	1	1	1	1	1
Autoren	1	1	1	1	1	1	1	1	1	1
Schlagworte	2	2	2	2	2	2	2	1 / 2	–	–
Jahr	1	1	1	1	1	1	1	1	1	1
Verlag	1	1	2	2	2	1	2	1 / 2	2	2
Auflage	2	2	2	2	2	2	–	–	–	–
Ausgabe	1	2	2	2	2	2	1	1 / 2	1	–
Publikationstyp	1	1	2	2	2	2	1	1 / 2	1	2
ÜbergWerk	–	–	–	–	–	–	1	1 / 2	1	2
Sprache	2	2	1	2	2	2	2	1 / 2	2	2
Seiten	2	2	2	2	2	2	2	1 / 2	1	2
Begleitmat.	1	1	1	2	2	2	–	–	–	–
ISXN	2	2	1	2	2	2	2	1 / 2	2	2
Preis	1	1	1	–	–	2	–	–	–	–
Titelbild	1	2	2	–	–	–	–	–	–	–
Inhaltsverz.	2-3	2	2	–	–	–	–	–	–	–
Beschreib.	2-3	2	2	–	–	–	1 / 2	1 / 2	–	2
Beurteilungen	1,2-3	2	–	–	–	–	2	–	–	–
Verkaufsrang	2	–	–	–	–	–	–	–	–	–
Zitierungen	–	–	–	–	–	–	2	–	–	2
Beschaffung	1	1	1	3-5	3-4	4	3	2 / 3	2	3

Tabelle 8.3: Übersicht über vertikale Informationsportionen und anfragbare Attribute

Für die Anfragebearbeitung eines Meta-Recherchesystems ist es neben der vertikalen Informationsportionierung auch von Bedeutung, an welche Attribute Anfragen gestellt werden können. Erlaubt das Meta-Recherchesystem beispielsweise die Formulierung von Bedingungen an Attribute, die bei den eingebundenen Diensten nicht anfragbar sind, so kann die Anfrage zunächst nur ohne diese Bedingung gestellt werden. Anschließend müssen dann die Dokumente herausgefiltert werden, die die Bedingung nicht erfüllen. Diese Aufgabe kann

zwar prinzipiell von Wandlern übernommen werden, allerdings hat dieses Vorgehen erhebliche Auswirkungen auf die Antwortzeit des Meta-Recherchesystems. Für derartige Anfragen müssen mehr Dokumente und mehr Informationen zu diesen Dokumenten beschafft werden, d.h. es müssen mehr HTTP-Aufrufe durchgeführt werden.

In Tabelle 8.3 sind die Attribute jedes Dienstes grau hinterlegt, die explizit angefragt werden können. RI erlaubt keine Anfragen an spezielle Attribute von Dokumenten, sondern nur die Suche im Volltext oder in den Zitierungen der Dokumente. CSBIB erlaubt keine beliebigen Anfragen an die Beschaffungsinformationen, sondern unterstützt lediglich die Einschränkung einer Anfrage auf online verfügbare Artikel.

8.3 Antwortzeitverhalten der Literaturdienste

In diesem Abschnitt werden Experimente entwickelt, durchgeführt und ausgewertet, um das Antwortzeitverhalten der ausgewählten Literaturdienste zu untersuchen. Dabei sollen zum einen allgemeine Aussagen über die zu erwartenden Antwortzeiten für die verschiedenen Informationsportionen gewonnen werden. Zum anderen sollen charakteristische Eigenschaften des Antwortzeitverhaltens sowie ggf. auch Abweichungen davon bei den betrachteten Literaturdiensten herausgearbeitet werden.

Um das Antwortzeitverhalten der ausgewählten Literaturdienste zu untersuchen, werden von einem Testrechner aus charakteristische Anfragen an die Dienste geschickt. Dabei wird die Zeit gemessen, die die Dienste zum Antworten benötigen, d.h. die Zeit, bis die Antwort beim Testrechner angekommen ist.

Die beobachtbare Antwortzeit setzt sich aus mehreren Teilen zusammen, beispielsweise aus der Netzübertragungszeit der Daten (Anfrage und Ergebnisse) und aus der Bearbeitungszeit der Anfrage im angesprochenen Server. Diese Bearbeitungszeit kann wiederum unterteilt werden, nämlich in die Bearbeitungszeit des Webservers und in die Bearbeitungszeit der Datenbank, welche den Dokumentbestand enthält. In der vorliegenden Arbeit wollen wir zur Aufschlüsselung der Antwortzeit keine technischen Zusatzinformationen heranziehen und damit auch keine detaillierteren Untersuchungen anstellen. In den Experimenten soll die Antwortzeit der Literaturdienste lediglich in der Weise berücksichtigt werden, wie sie auch von einem Benutzer oder von einem Meta-Recherchesystem wahrgenommen wird.

Definition 8.1: Unter der *Antwortzeit* eines Literaturdienstes verstehen wir die Zeit vom Absenden einer Anfrage bzw. eines HTTP-Aufrufs von einem Testrechner bis zum Eintreffen des Ergebnisses bzw. der HTML-Seite mit dem Ergebnis auf diesem Testrechner. Die Zeitmessung berücksichtigt die Zeitspanne, bis die empfangene Informationsportion vollständig auf dem Testrechner angekommen ist (time to last byte).

Bei den folgenden Experimenten wird die Antwortzeit jeweils in Millisekunden (ms) gemessen und ausgewiesen. Der Testrechner ist ein Notebook vom Typ Dell Latitude C 600,

welcher mit einer Übertragungsrate von 10 MBit/s über das LAN der Informatik-Fakultät Karlsruhe auf das Internet zugreift.

8.3.1 Experiment 1 – Allgemeiner Überblick

Im ersten Experiment soll ein Überblick über die durchschnittlichen Werte und das Spektrum der Antwortzeiten bei den ausgewählten Literaturdiensten gewonnen werden.

8.3.1.1 Aufbau und Durchführung

Literaturdienste erlauben im Wesentlichen zwei Arten von Anfragen: Zum einen die wirklichen Anfragen, die Suchbegriffe und Bedingungen enthalten und als Ergebnis eine Kurztitelliste liefern, und zum anderen Anforderungen von zusätzlichen Informationsportionen zu einem Dokument. Zur Unterscheidung wollen wir diese Informationsanforderungen im Folgenden Link-Expansionen nennen, da dabei i.d.R. eine URL verfolgt wird, die in einer vorangegangenen Informationsportion angegeben wurde. Um das Antwortzeitverhalten eines Literaturdienstes zu ermitteln, müssen beide Arten von Anfragen berücksichtigt werden.

Anfragen erzeugen als Ergebnis eine HTML-Seite, die einen Teil der gefundenen Treffer im Kurztitelformat enthält. Für eine Anfrage wollen wir zum einen die Anzahl der im Dokumentbestand gefundenen Treffer berücksichtigen und zum anderen die Anzahl der Treffer, die davon auf der Ergebnisseite präsentiert werden. Im folgenden Experiment soll das Antwortzeitverhalten die Dienste anhand von unterschiedlich großen Anfragen und Link-Expansionen auf verschiedenen Informationsebenen ausgetestet werden.

Definition 8.2: Die *Größe* einer Anfrage bezeichnet die Anzahl der in einem Dokumentbestand gefundenen Treffer zu der gegebenen Anfrage.

Um die Antwortzeiten von unterschiedlichen Literaturdiensten miteinander vergleichen zu können, müssen solche Anfragen an die Dienste gestellt werden, die gemäß des jeweiligen Dokumentbestands in etwa dieselbe Größe haben (+/- 10%). Um aussagekräftige Ergebnisse zu gewinnen, werden für diverse Anfragegrößen bei jeden Dienst fünf entsprechende Anfragen zusammengestellt. Für jede Informationsebene mit Zusatzinformationen zu einem Dokument werden ebenfalls jeweils fünf verschiedene Link-Expansionen für jeden Dienst ausgewählt.

Jede Anfrage bzw. jede Link-Expansion wird nun in regelmäßigen Abständen über einen gewissen Zeitraum getestet, um aussagekräftige Durchschnittswerte zu erhalten. Für das vorliegende Experiment wurde jede Anfrage bzw. jede Link-Expansion mehr als 500 Mal in der Zeit von einer Woche (6. - 13. November 2001) ausgeführt. D.h. jede Anfrage bzw. jede Link-Expansion wurde etwa alle 20 Minuten an den entsprechenden Dienst geschickt.

Um nun die durchschnittlichen Antwortzeiten einer gegebenen Anfragegröße bzw. einer gegebenen Stufe der Link-Expansion zu ermitteln, können verschiedene Berechnungen

durchgeführt werden. Beispielsweise kann der Mittelwert über alle Antwortzeiten der fünf verschiedenen Anfragevarianten ermittelt werden. Um nicht die Antwortzeiten von unterschiedlichen Anfragen miteinander zu vermengen, wird zur Auswertung des vorliegenden Experiments ein anderes Verfahren gewählt: Zunächst wird von jeder der fünf Anfragevarianten der Mittelwert gebildet. Anschließend wird diejenige Anfragevariante als repräsentativ ausgewählt und berücksichtigt, deren Mittelwert den Median der fünf gebildeten Mittelwerte darstellt.

8.3.1.2 Ergebnisse

Bibliotheken. Zuerst betrachten wir die Klasse der Bibliotheksdienste. Für die Experimente werden unterschiedlich große Anfragen und unterschiedlich große Ergebnisportionen verwendet. Um vergleichbare Resultate zu erzielen, muss bei dem UBKA- und dem BLB-Dienst die Anzahl der angezeigten Treffer auf 10 reduziert werden, da der DBF-Dienst die Ergebnistreffer ausschließlich in 10-er-Portionen anbietet (vgl. Tabelle 8.2). Für die Experimente werden Anfragen mit 10, 30 und 200 Ergebnissen berücksichtigt, von denen entweder 10 oder 30 Treffer in Form von Kurztiteln präsentiert werden. Darüber hinaus werden Link-Expansionen der Ebenen 2 (Bibliographische Informationen) und 3 (Beschaffungsinformationen) getestet. Da für die Experimente kein Benutzerzugang für den DBF-Dienst zur Verfügung stand, konnten die Antwortzeiten von Ebene 3 dort nicht berücksichtigt werden. Tabelle 8.4 gibt eine Übersicht über die durchschnittlichen Antwortzeiten der genannten Anfragearten.

	UBKA	BLB	DBF
10 Kurztitel (von 10 Ergebnissen)	962	2740	461
30 Kurztitel (von 30 Ergebnissen)	1162	3234	–
10 Kurztitel (von 200 Ergebnissen)	1405	3898	521
30 Kurztitel (von 200 Ergebnissen)	1693	4644	–
Bibliographische Informationen (Ebene 2)	1717	1367	341
Beschaffungsinformationen (Ebene 3)	2806	4705	–

Tabelle 8.4: Durchschnittliche Antwortzeiten der Bibliotheken (in Millisekunden)

Die Antwortzeiten der Bibliotheksdienste sind recht unterschiedlich. Der DBF-Dienst antwortet besonders schnell. Er liefert seine Ergebnisse in weniger als einer Sekunde und das, obwohl er mehrere Hundert Kilometer von Karlsruhe entfernt ist und die beiden anderen Dienste sowie der Testrechner in Karlsruhe lokalisiert sind. Auch zwischen den Bibliotheken vor Ort existieren große Unterschiede. Der UBKA-Dienst liefert seine Ergebnisse etwa doppelt so schnell wie der BLB-Dienst.

Innerhalb eines Bibliotheksdienstes lässt sich folgende Korrelation beobachten: Die Antwortzeit erhöht sich deutlich bei Anfragen, bei denen eine größere Anzahl an Ergebnissen

im Dokumentbestand gefunden wird, auch wenn schließlich dieselbe Anzahl von Treffern auf der Ergebnisseite präsentiert wird. Aber auch die Anzahl der präsentierten Treffer wirkt sich auf die Antwortzeit aus, wobei sich die Antwortzeit bei mehr präsentierten Treffern in jedem Fall unterproportional verlängert.

Die Antwortzeiten der zweiten und dritten Informationsebene weisen kein klares Muster auf. Im allgemeinen können die bibliographischen Informationen schneller bezogen werden als die Ausleihinformation. Vom UBKA- und vom BLB-Dienst ist beispielsweise bekannt, dass diese zwei separate Datenbasen für die bibliographischen Daten und die aktuellen Ausleihinformationen betreiben, und diese können natürlich unterschiedliche Leistungsmerkmale haben.

Buchhändler. Der zweite Teil des Experiments beschäftigt sich mit den Buchhändlern. Auch hier werden wiederum unterschiedlich große Anfragen und unterschiedlich große Ergebnisportionen untersucht. Zwei der Dienste (AMAZON und KNO&KV) bieten die Möglichkeit an, eine andere Ergebnissortierung als die vorgegebene Standardeinstellung zu wählen. Dies soll ebenso in den Experimenten berücksichtigt werden wie die Option, alle Kurztitel auf einer Ergebnisseite anzuzeigen. Diese Option boten zum Zeitpunkt der Experimente zwei Dienste an (AMAZON und LOB), allerdings unterstützt AMAZON diese Funktionalität mittlerweile nicht mehr (vgl. Tabelle 8.2). Auch bei den Buchhändlern werden wieder Link-Expansionen getestet, allerdings nur für die zweite Ebene der bibliographischen Informationen und der inhaltlichen Zusatzinformationen. Tabelle 8.5 liefert eine Übersicht über die durchschnittlichen Antwortzeiten der genannten Anfragearten.

	AMAZON	KNO&KV	LOB
10 Kurztitel (von 10 Ergebnissen)	2609	1049	978
30 Kurztitel (von 30 Ergebnissen)	3714	1849	1070
30 Kurztitel (von 30 Ergebnissen; andere Sortierung)	3505	1691	–
30 Kurztitel (von 100 Ergebnissen)	3484	2038	1110
30 Kurztitel (von 100 Ergebnissen; andere Sortierung)	3768	2310	–
100 Kurztitel (von 100 Ergebnissen)	8400	–	1329
100 Kurztitel (von 500 Ergebnissen)	8828	–	1503
Bibliographische Informationen (Ebene 2)	1841	830	966

Tabelle 8.5: Durchschnittliche Antwortzeiten der Buchhändler (in Millisekunden)

Obwohl Buchhändler relativ große Dokumentbestände verwalten, sind die Antwortzeiten von KNO&KV und LOB im Vergleich zu denen der Bibliotheksdienste UBKA und BLB recht schnell. Vermutlich stehen die kommerziellen Dienste auch in einem direkteren Konkurrenzverhältnis zueinander und legen daher mehr Gewicht auf die Benutzerzufriedenheit, was sich

in attraktiveren Informationsportionierungen, Sortierfunktionalitäten und eben auch in schnellen Antwortzeiten bemerkbar macht. Bei größeren Ergebnismengen skaliert LOB deutlich besser als AMAZON. Die Antwortzeiten von Informationsportionen der Ebene 2 lassen sich auch hier nur schwer in Zusammenhang mit den anderen Antwortzeiten des jeweiligen Dienstes bringen. Bei den Buchhändlern können die bibliographischen Daten zu einem Dokument i.d.R. schneller geliefert werden als die Kurztitelliste einer sehr kleinen Anfrage.

Die Korrelation, die bei den Bibliotheksdiensten beobachtet wurde, trifft auch hier zu. Sowohl die Größe der Ergebnismenge als auch die Menge der davon angezeigten Kurztitel haben Auswirkungen auf die Antwortzeit. Lediglich die Antwortzeiten von AMAZON mit der Standardsortierung scheinen der allgemeinen Beobachtung zu widersprechen, wohingegen die Antwortzeiten mit einer anderen Sortierung die Beobachtung wiederum unterstützen. Die Experimente zu unterschiedlichen Ergebnissortierungen weisen inkonsistente Effekte im Hinblick auf die Antwortzeit auf. Vermutlich sind umfangreichere und detailliertere Experimente zu den Sortierfunktionen notwendig, um aussagekräftige Resultate erzielen zu können. Darauf soll im Rahmen der vorliegenden Arbeit allerdings verzichtet werden.

Artikel-Dienste. Der dritte Teil des Experiments beschäftigt sich mit den Literaturdiensten, die hauptsächlich wissenschaftliche Artikel zur Verfügung stellen. Auch hier werden wiederum unterschiedlich große Anfragen und unterschiedlich große Ergebnisportionen untersucht. Zwei Dienste (CSBIB und ACM-DL) bieten die Option an, die Kurztitel mit weniger und mit mehr Informationen zu liefern. Diese beiden Varianten sollen in den Experimenten berücksichtigt und miteinander verglichen werden. Drei Dienste unterstützen Ergebnisportionen mit einer größeren Anzahl von Kurztiteln (CSBIB, DBLP und RI), wobei RI mittlerweile nur noch maximal 50 Kurztitel auf einer Seite präsentiert (vgl. Tabelle 8.2). Auch im Rahmen dieser Experimente werden Link-Expansionen getestet, sie beziehen sich bei den Artikel-Diensten auf die zweite Ebene der bibliographischen Informationen. Tabelle 8.5 gibt eine Übersicht über die durchschnittlichen Antwortzeiten der genannten Anfragearten.

	ACM-DL	CSBIB	DBLP	RI
20 Kurztitel (von 20 Ergebnissen) kurz	1359	1650	197	1688
20 Kurztitel (von 100 Ergebnissen) kurz	1372	1670	270	2442
20 „lange“ Kurztitel (von 20 Ergebnissen)	1445	2958	–	–
20 „lange“ Kurztitel (von 100 Ergebnissen)	1396	3105	–	–
100 Kurztitel (von 100 Ergebnissen)	–	3646	328	5136
Bibliographische Informationen (Ebene 2)	1366	402	–	1471

Tabelle 8.6: Durchschnittliche Antwortzeiten der Artikel-Dienste (in Millisekunden)

In dieser Dienstklasse gibt es wieder einen sehr schnellen Dienst (DBLP), der einige Hundert Kilometer vom Testrechner entfernt angesiedelt ist und der deutlich schneller antwortet als ein Dienst, der ebenfalls in Karlsruhe angeboten wird (CSBIB). Selbst der in USA ansässige Dienst ACM-DL liefert schnellere Antwortzeiten als CSBIB, so dass die Beobachtung abgeleitet werden kann, dass die „Länge“ der Übertragungswege im Internet insgesamt keine dominierende Rolle spielen kann.

Bei ACM-DL macht sich die zusätzlich angezeigte Zusammenfassung jedes Artikels nicht in der Antwortzeit bemerkbar, bei CSBIB ist eine Verlangsamung bei der ausführlichen Trefferliste zu bemerken, allerdings werden dabei auch sämtliche Attribute zu jedem Dokument aufgeführt. Auch bei den Artikel-Diensten hat die Größe der Anfrage und die Anzahl der präsentierten Treffer einen Einfluss auf die Antwortzeit.

8.3.1.3 Diskussion und weitere Beobachtungen

Obwohl die durchschnittlichen Antwortzeiten stark zwischen den unterschiedlichen Literaturdiensten und unterschiedlichen Arten von Anfragen variieren, bleiben sie doch alle in einem gewissen Bereich. Alle Antwortzeiten liegen unter 10 Sekunden, die meisten sogar unter 5 Sekunden.

Es gibt gesicherte Werte über akzeptable Antwortzeiten für Benutzer [Nie94]. Dabei existieren drei wesentliche Zeitgrenzen für die Reaktion des Systems: 1/10 Sekunde ist die Zeitdauer, bei der der Benutzer den Eindruck hat, dass das System sofort reagiert. 1 Sekunde ist die Zeitdauer, in der der Gedankengang des Benutzers noch nicht unterbrochen wird, obwohl der Benutzer die Verzögerung bemerkt. 10 Sekunden ist die Zeitdauer, in der der Benutzer mit seiner Aufmerksamkeit beim Dialog mit dem System bleibt. Bei längeren Verzögerungen beginnt der Benutzer andere Tätigkeiten, während er auf die Antwort des Systems wartet.

Die 10-Sekunden-Grenze wird von allen betrachteten Literaturdiensten eingehalten. D.h. die Antwortzeiten der ausgewählten Dienste sind für die direkte Nutzung durch einen Benutzer durchaus akzeptabel. Das trifft sicherlich in einem noch größeren Maße zu, wenn bei der Bestimmung der Antwortzeiten zusätzlich auch die Zeit bis zum Eintreffen der ersten Ergebnisse ausgewiesen werden würde (time to first byte). Inwiefern die ermittelten Antwortzeiten nun auch für die Nutzung im Rahmen eines Meta-Recherchesystems akzeptabel sind, wird im weiteren Verlauf der Arbeit behandelt.

Zusätzlich zu den im vorangegangenen Abschnitt geschilderten Ergebnissen wurden auch die Ausreißer der Antwortzeiten ausgewertet, d.h. die extrem langen Antwortzeiten, die im Web-Umfeld bekanntermaßen häufig anzutreffen sind. Dabei haben wir einen Ausreißer als eine Antwortzeit von mehr als 10 Sekunden definiert. Bis auf BLB und AMAZON konnten die betrachteten Dienste mindestens 99 % der gestellten Anfragen bzw. Link-Expansionen in weniger als 10 Sekunden beantworten. Bei BLB und AMAZON dauerten etwa 5 % der

Anfragen länger als 10 Sekunden, wobei diese Werte bei AMAZON auch nicht alle zwingend als Ausreißer eingestuft werden dürfen, da sie teilweise durchaus im normalen Toleranzbereich von durchschnittlichen Antwortzeiten von ca. 8 Sekunden liegen.

Weiterhin wurde die Zuverlässigkeit der Literaturdienste untersucht. Auch hier ist das Ergebnis sehr positiv: Im Zeitraum der Experimente lieferten sämtliche Dienste in weniger als 1 % der Fälle Fehlermeldungen, was bedeutet, dass die Fehlerquote statistisch gesehen nicht relevant ist.

Neben dem Durchschnittswert haben wir auch die Standardabweichung für jede Anfrage bzw. jede Link-Expansion bei jedem Dienst berechnet. Die Standardabweichung beschreibt die mittlere Abweichung der Antwortzeiten von der berechneten Durchschnittsantwortzeit. Sämtliche ermittelten Standardabweichungen sind ziemlich hoch, sie weisen Werte von 1000 bis 5000 (und teilweise auch noch mehr) auf. Das kann so interpretiert werden, dass die Antwortzeiten durchschnittlich etwa 1 bis 5 Sekunden vom berechneten Mittelwert abweichen. Für jeden einzelnen Literaturdienst sind die ermittelten Standardabweichungen nahezu konstant, d.h. unabhängig von der Art der getesteten Anfrage. Die Dienste mit den schnellsten Antwortzeiten haben auch die kleinsten Standardabweichungen (DBF, LOB und DBLP), wohingegen BLB und AMAZON Standardabweichungen von mehr als 5000 aufweisen.

Für die hohen Standardabweichungen sind verschiedene Ursachen denkbar. Möglicherweise ist die Anzahl der getesteten Anfragen zu gering, möglicherweise folgen die Antwortzeiten keiner einheitlichen Verteilung oder möglicherweise haben auch die Tageszeiten einen Einfluss auf die unterschiedlichen Antwortzeiten.

Nachdem das geschilderte Experiment nun erste Aussagen über das allgemein zu beobachtende Antwortzeitverhalten der Literaturdienste erlaubt, sollen in einem zweiten Experiment weniger Anfragen getestet werden, diese aber dafür in einem wesentlich umfangreicheren Ausmaß, so dass dadurch Hinweise bzw. Erklärungen für die hohen Standardabweichungen gefunden zu können.

8.3.2 Experiment 2 – Spezielle Vertiefung

Mit dem zweiten Experiment sollen zwei Absichten verfolgt werden: Zum einen soll die Verteilung der Antwortzeiten genauer analysiert werden und zum anderen sollen die Antwortzeiten im Hinblick auf Abhängigkeiten von der Tageszeit untersucht werden. Dazu sollen einige ausgewählte Anfragen eingehender, d.h. häufiger und über einen längeren Zeitraum hinweg, getestet werden.

Für das zweite Experiment werden sechs repräsentative Anfragen des vorangegangenen Experiments ausgewählt: Es werden drei Dienste (UBKA, BLB und AMAZON) jeweils mit einer mittelgroßen Anfrage und einer Link-Expansion der zweiten Ebene getestet. Das zweite

Experiment wurde über einen Zeitraum von 3 Wochen (im September 2001) durchgeführt, dabei wurden pro Anfrage mehr als 130.000 Antwortzeiten ermittelt.

Die Ergebnisse des zweiten Experiments können ausführlich in [SO02a] und [Obe02] nachgelesen werden. Um den Rahmen der vorliegenden Arbeit nicht zu sprengen, greifen wir in diesem Abschnitt nur eine der sechs Anfragen für eine detaillierte Betrachtung auf. Die Verteilung der Antwortzeiten zu dieser Anfrage kann als repräsentativ für die meisten der betrachteten Anfragen an Literaturdienste gelten. In die abschließende Diskussion der Ergebnisse werden noch einige Beobachtungen der anderen Auswertungen mit einfließen.

8.3.2.1 Zugriffsstatistik

Einige Literaturdienste veröffentlichen eine Zugriffsstatistik ihres Webservers. Diese Statistiken geben i.d.R. an, wie viele Zugriffe pro Monat, pro Tag oder sogar pro Stunde erfolgen. Besonders wenn ein Literaturdienst größtenteils von lokalen Benutzern oder Benutzern im selben Land (und damit auch in derselben Zeitzone) genutzt wird, variiert die Anzahl der Zugriffe stark in Abhängigkeit der jeweiligen Tageszeit. Wir gehen davon aus, dass die Zahl der Zugriffe auf den Webserver im Wesentlichen auch der Zahl der Anfragen und der Link-Expansionen entspricht. Abbildung 8.1 zeigt die Anzahl der Zugriffe auf den Webserver des UBKA-Dienstes pro Monat (September 2001), aufgeschlüsselt nach den jeweiligen Zeitstunden (vgl. http://www.ubka.uni-karlsruhe.de/statistik/).

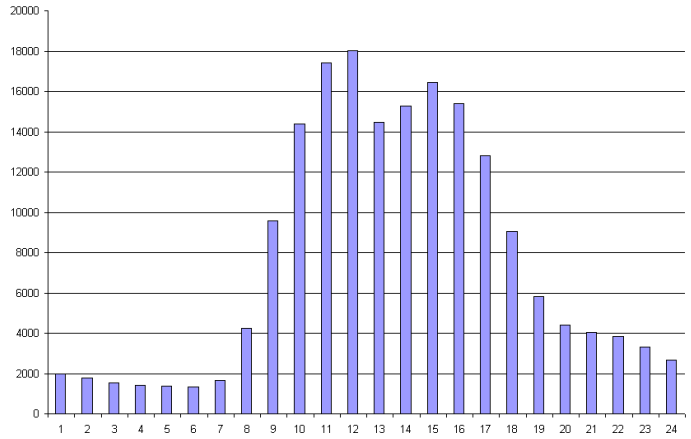


Abbildung 8.1: Monatliche Benutzerstatistik von UBKA, aufgegliedert nach Stunden

Die meisten Anfragen werden in den Mittags- und Nachmittagsstunden gestellt. Im folgenden Experiment soll nun untersucht werden, ob die große Varianz der Antwortzeiten auf die verschiedenen Tageszeiten zurückgeführt werden kann.

8.3.2.2 Ergebnisse

Stellvertretend für das zweite Experiment soll nun die UBKA-Anfrage (mit 30 Ergebnissen und 30 Kurztiteln) genauer betrachtet werden. Die durchschnittliche Antwortzeit der getesteten Anfrage beträgt 1035 ms. Im ersten Experiment wurde für dieselbe Anfrage eine Antwortzeit von 1162 ms ermittelt. Dass beide Werte sehr nah beieinander liegen, unterstützt auch noch einmal die Aussagefähigkeit von Experiment 1, auch wenn die Durchschnittswerte dabei aus deutlich weniger Messwerten berechnet wurden. Die Standardabweichung ist mit einem Wert von 432 im zweiten Experiment nun wesentlich geringer als zuvor (1569), kann allerdings auch nicht vernachlässigt werden. Immerhin bedeutet dieser Wert, dass die Antwortzeiten durchschnittlich um eine halbe Sekunde vom berechneten Mittelwert abweichen.

Um die Verteilung der Antwortzeiten anschaulich darzustellen, wird ein Histogramm erstellt, welches im oberen Teil von Abbildung 8.2 zu sehen ist. Das Histogramm weist einen relativ klaren Kurvenverlauf auf. Die meisten Testanfragen konnten sehr schnell beantwortet werden, das zeigt sich an der spitzen Erhebung im Bereich von 800 bis 950 ms. Der Rest der Antwortzeiten liegt im Wesentlichen im Bereich zwischen 950 und 1900 ms. Andere, d.h. längere Antwortzeiten kommen dagegen vergleichsweise selten vor.

Der Kurvenverlauf liefert gleichzeitig auch die Erklärung für die ja immer noch recht hohe Standardabweichung. Da die durchschnittliche Antwortzeit deutlich neben der spitzen Erhebung im linken Bereich liegt, weicht schon einmal die Masse dieser Antwortzeiten erheblich von dem Mittelwert ab, genauso wie die Antwortzeiten, die sozusagen in einer flachen Stufe rechts neben der Spitze und dem Mittelwert angesiedelt sind.

Im unteren Teil von Abbildung 8.2 sind die ermittelten Antwortzeiten durch kleine schwarze Punkte dargestellt. Die Antwortzeiten sind über einer Zeitachse gemäß der Zeitstunden des Tages aufgetragen, in denen sie gemessen wurden. Anhand der Farbschattierungen, die sich aus der Verteilung der Anfragezeiten ergeben, kann man sehen, dass in den Mittags- und Nachmittagsstunden etwas weniger Antwortzeiten zur spitzen Erhebung der Kurve und dafür mehr Antwortzeiten zur anschließenden Stufe beitragen. Eine Berechnung der durchschnittlichen Antwortzeiten pro Stunde ergibt für die Tagesstunden einen Wert von etwa 1100 ms, in der Nacht beträgt die durchschnittliche Antwortzeit ungefähr 950 ms. In Anbetracht der enormen Unterschiede in den Zugriffszahlen der Benutzer ist dieser Unterschied eher als gering zu werten.

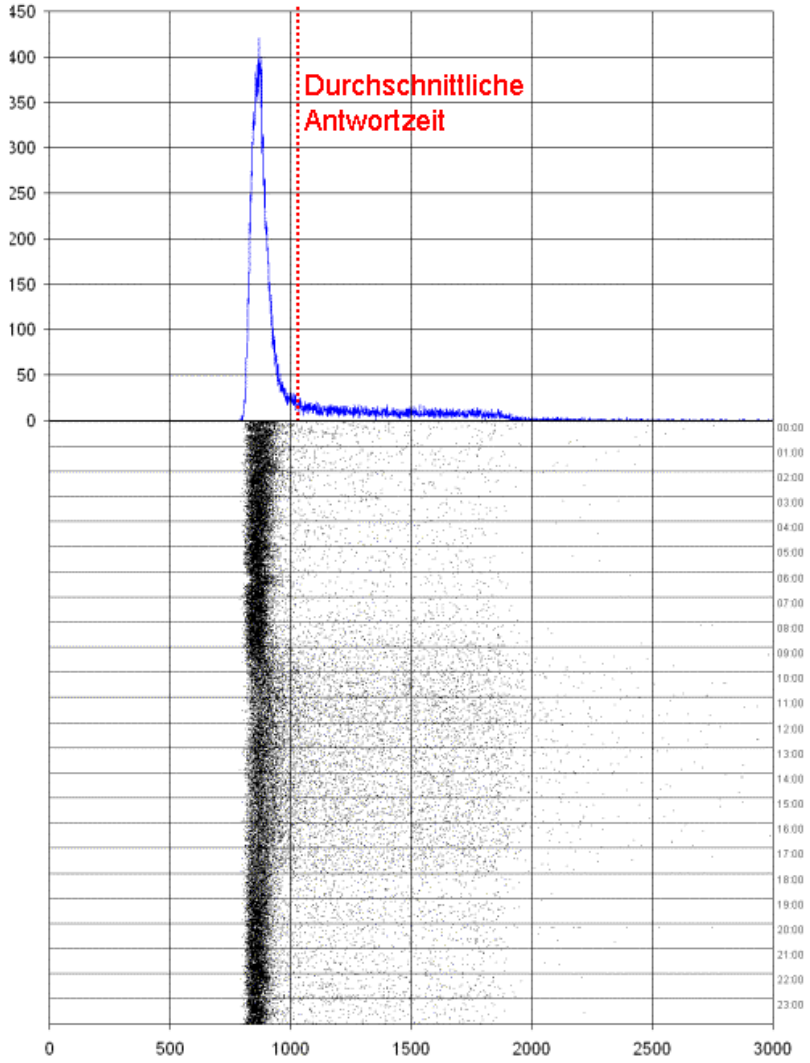


Abbildung 8.2: Histogramm der Antwortzeiten, aufgeschlüsselt nach Stunden

8.3.2.3 Diskussion und weitere Beobachtungen

Der im vorangegangenen Abschnitt ermittelte Kurvenverlauf der Anfrageverteilung ist repräsentativ für die Mehrheit der untersuchten Anfragen. Die meisten Anfrageverteilungen weisen eine Spitze (links des Mittelwerts) mit anschließender Stufe (rechts des Mittelwerts) auf, sie unterscheiden sich lediglich in der Ausprägung, d.h. in der Breite und Höhe der Spitze bzw. der Stufe.

Betrachtet man den Webserver, der die Anfragen der Benutzer bzw. unsere Testanfragen bearbeitet, lässt sich der gefundene Kurvenverlauf etwa folgendermaßen erklären: In der meisten Zeit kann der Webserver die Anfragen recht zügig bearbeiten. Natürlich gibt es zu jeder Anfrage eine minimale Antwortzeit, die aufgrund des Datenübertragungswegs und einer gewissen Bearbeitungszeit im Literaturdienst nicht unterschritten werden kann. Im vorliegenden Experiment liegt beispielsweise keine Antwortzeit unter 800 ms. Die meisten Antwortzeiten befinden sich allerdings nahe der minimalen Antwortzeit, da die Kurve sozusagen sofort mit einer steilen Erhebung beginnt.

Ein Webserver verwaltet eine gewisse Anzahl von parallelen Verbindungen. Wenn es mehr Anfragen als parallele Verbindungen gibt, werden die überzähligen Anfragen in eine Warteschlange eingereiht. Die Anfragen in der Warteschlange werden dann abgearbeitet, sobald wieder Verbindungen frei geworden sind. Damit liegt die Vermutung nahe, dass die längeren Antwortzeiten aufgrund von zusätzlichen Wartezeiten in der Schlange zustande kommen. Gerade in der Mittags- und Nachmittagszeit, wo von den Benutzern so viele Anfragen gestellt werden, ist die Anzahl der parallelen Verbindungen natürlich schneller ausgeschöpft als in den Nachtstunden.

Der Webserver des UBKA-Dienstes unterstützt beispielsweise 100 parallele Verbindungen. Daraus dass auch tagsüber die durchschnittlichen Antwortzeiten nicht wesentlich langsamer werden, kann gefolgert werden, dass der Webserver für die von den Benutzern erzeugte Last geeignet dimensioniert ist.

Eine Ausnahme zu dem gefundenen Antwortzeitverhalten stellt AMAZON dar. Das Histogramm der Antwortzeiten von AMAZON zeigte sowohl bei der getesteten Anfrage als auch bei der getesteten Link-Expansion extrem uneinheitliche Verläufe mit mehr als fünf Spitzen. Hier brachten die Aufschlüsselungen nach den Tageszeiten entscheidende Hinweise für die Interpretation: Zu bestimmten Zeitpunkten am Tag wechselten sich die häufigsten Antwortzeiten (die dann zu größeren oder kleineren Spitzen im Kurvenverlauf führten) abrupt ab. Das legt die Vermutung nahe, dass AMAZON spezielle Lastregulierungsmechanismen verwendet und damit auf die unterschiedliche Benutzerlast reagiert. Unter den betrachteten Literaturdiensten stellt AMAZON damit einen Einzelfall dar. Die anderen Literaturdienste folgen dem bei UBKA beobachteten Antwortzeitverhalten.

In Bezug auf Ausreißer in den Antwortzeiten und die Verfügbarkeit der Dienste können die Beobachtungen des ersten Experiments bestärkt werden, im zweiten Experiment wurden prozentual sogar noch weniger Ausreißer (0,53%) und Ausfälle (<0,01%) ermittelt.

8.4 Formale Beschreibung des Antwortzeitverhaltens

Die Experimente des vorangegangenen Abschnitts konnten einige Aufschlüsse über das Antwortzeiten existierender Literaturdienste liefern. In vorliegenden Kapitel sollen Literaturdienste in ihrer Struktur und in ihrem Antwortzeitverhalten beschrieben und in einem späteren Teil der Arbeit (Kapitel 12 und 13) dann auch simuliert werden. Die Experimente haben gezeigt, dass ein naiver Ansatz, der etwa die Antwortzeiten eines Literaturdienstes durch einen Durchschnittswert beschreibt, nicht der Realität entspricht. Vielmehr konnte ein charakteristisches Antwortzeitverhalten identifiziert werden: Ein Großteil der Antwortzeiten liegt unter dem ermittelten Durchschnittswert (Spitze), die anderen Antwortzeiten bleiben in einem gewissen Rahmen oberhalb des Durchschnittwerts (Stufe). Die Variationen der Literaturdienste zeigen sich in unterschiedlichen Ausprägungen und Dimensionierungen der Spitze und der Stufe.

Der identifizierte Kurvenverlauf soll in diesem Abschnitt formal beschrieben werden. Dazu soll eine möglichst ähnliche Wahrscheinlichkeitsverteilung gefunden und ggf. modifiziert werden. Damit kann die Verteilung der Antwortzeiten anhand der entsprechenden Parameter der Wahrscheinlichkeitsverteilung und ggf. zusätzlicher Parameter beschrieben werden.

Um eine geeignete Annäherung der Histogrammkurve durch eine Wahrscheinlichkeitsverteilung zu bestimmen, werden zunächst einige Wahrscheinlichkeitsverteilungen zusammengestellt, deren Kurvenverläufe eine gewisse Ähnlichkeit mit dem gefundenen Antwortzeitverhalten aufweisen [BS00]. Tabelle 8.7 zeigt drei unterschiedliche Wahrscheinlichkeitsverteilungen bzw. den Verlauf ihrer Dichtefunktionen und deutet die Veränderlichkeit der jeweiligen Kurvenform aufgrund von gewählten Parameterbelegungen an.

Betrachtet man die dargestellten Kurvenverläufe, so ist die Dichte der Weibull-Verteilung dem gegebenen Antwortzeitverlauf insofern am ähnlichsten, als die Spitze auf der linken Seite auch relativ steil beginnen kann. Die anderen beiden Dichtekurven steigen dagegen eher langsam an. Eine Stufe schließt bei keiner der dargestellten Kurven an die Spitze an. Allerdings kann eine solche Stufe auch durch Hinzuaddieren einer entsprechenden Gleichverteilung ergänzt werden.

Im Hinblick auf die mathematische Handhabbarkeit ist die Weibull-Verteilung ebenfalls den anderen beiden Verteilungen vorzuziehen, da sich recht leicht die inverse Funktion (f^{-1}) berechnen lässt, die im folgenden Verlauf für die Generierung einer entsprechenden Stichprobe benötigt wird. Für weitere Details dazu siehe [Obe02].

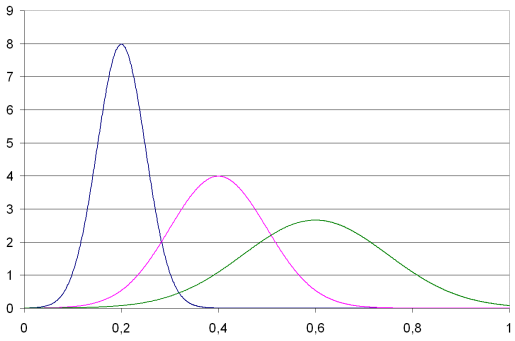
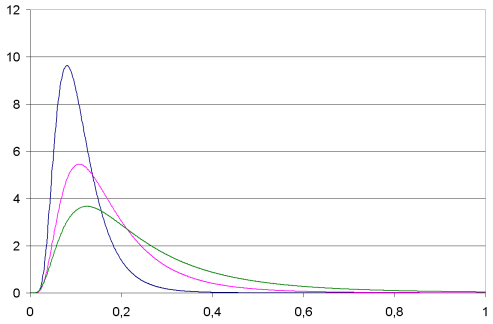
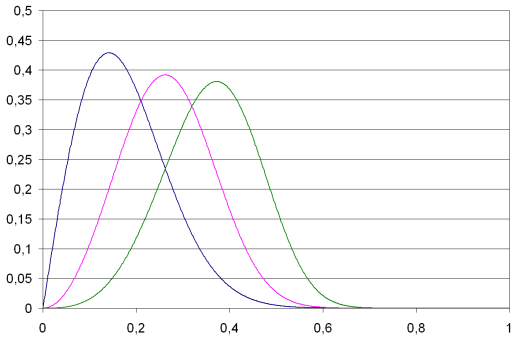
Wahrscheinlichkeitsverteilung	Beispielverläufe
Normalverteilung Dichte: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	
Lognormalverteilung Dichte: $f(x) = \begin{cases} 0 & , \quad x \le 0 \\ \frac{\log e}{x\sigma\sqrt{2\pi}} e^{-\frac{(\log x-\mu)^2}{2\sigma^2}} & , \quad x > 0 \end{cases}$	
Weibull-Verteilung Dichte: $f(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{(\alpha-1)} e^{-\left(\frac{x}{\beta}\right)^{\alpha}}$	

Tabelle 8.7: Unterschiedliche Wahrscheinlichkeitsverteilungen

Um die Eignung der Weibull-Verteilung in Kombination mit einer Gleichverteilung zu überprüfen, soll der in Abschnitt 8.3.2.2 gefundene Verlauf der Antwortzeiten mit einer generierten Stichprobe gemäß der genannten Verteilungen angenähert werden. Dazu werden die Beträge der Häufigkeiten der Antwortzeiten zunächst normiert, d.h. jeder Häufigkeitswert wird durch die Anzahl der Messwerte geteilt. Dadurch entsteht sozusagen eine diskrete Wahrscheinlichkeitsdichte. Das Ergebnis ist als blauer[12] bzw. dunkler Kurvenverlauf in

[12] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

Abbildung 8.3 zu sehen. Weiterhin zeigt nun die Abbildung die Annäherung dieser Kurve durch eine generierte Stichprobe (roter bzw. heller Kurvenverlauf) gemäß der genannten Verteilungen [Obe02]. Eine derartig erzeugte Stichprobe ist demnach durchaus geeignet, um das Antwortzeitverhalten eines Dienstes zu einer gegebenen Anfrage zu simulieren.

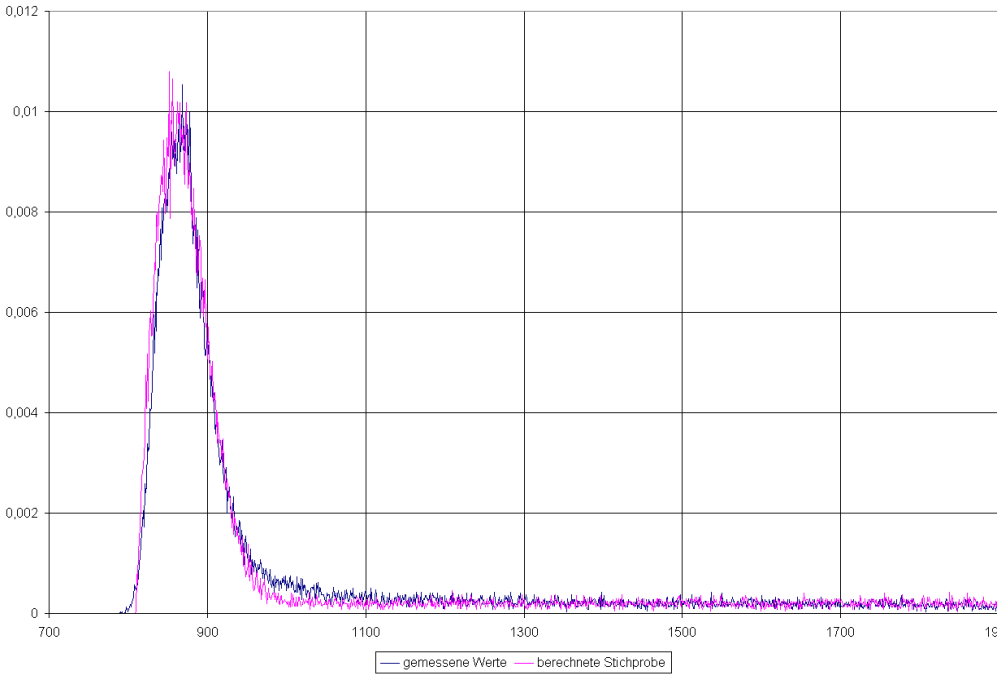


Abbildung 8.3: Ermitteltes Antwortzeitverhalten und angepasste Stichprobe

Die formale Beschreibung eines Antwortzeitverlaufs kann nun über die Angabe der entsprechenden Parameter α und β der Weibull-Dichte sowie der entsprechenden Parameter der Gleichverteilung zur Beschreibung der zusätzlichen Stufe erfolgen. Damit ist das Antwortzeitverhalten so präzise definiert, dass daraus eine entsprechende Stichprobe generiert werden kann. Da sich diese Parameter allerdings nicht direkt im Kurvenverlauf widerspiegeln, ist es für einen menschlichen Betrachter nahezu unmöglich, eine gewünschte Antwortzeitverteilung anhand von diesen Parametern zu spezifizieren bzw. sich aufgrund von gegebenen Parametern den entsprechenden Kurvenverlauf vorstellen zu können.

Daher wird zusätzlich ein anderer Ansatz verfolgt. Die Antwortzeitverteilung soll durch Parameter beschrieben werden, die einem menschlichen Betrachter ins Auge fallen. Aus diesen Angaben können dann die entsprechenden Parameterbelegungen von α, β und den zusätzlichen Parametern berechnet werden.

Folgende Parameter bieten sich zur Beschreibung des charakteristischen Kurvenverlaufs an:

- Minimale Antwortzeit: Das ist die Zeitschranke, die von keiner Antwortzeit unterlaufen werden kann.
- Maximale Antwortzeit: Das beschreibt das Ende der konstanten Stufe. Ausreißer werden also bei der Beschreibung des Kurvenverlaufs nicht berücksichtig.
- Breite der Spitze: Die Angabe bezieht sich auf die Distanz zwischen der minimalen Antwortzeit und dem Wendepunkt der Kurve von der abfallenden Spitze.
- Anteil der Spitze: Die Angabe bezieht sich auf die Fläche unter der Kurve, zusammen mit der maximalen Antwortzeit kann damit die Höhe der anschließende Stufe berechnet werden.

Mit diesen Parametern kann das Antwortzeitverhalten eines Dienstes bei einer bestimmten Art von Anfrage beschrieben werden. Dabei sind die Ausreißer in den Antwortzeiten noch nicht berücksichtigt. Sie sind allerdings notwendig, um eine realistische Simulation des Antwortzeitverhaltens durchzuführen. Daher benötigen wir noch einen zusätzlichen Parameter:

- Prozentualer Anteil von Ausreißern: Zur Beschreibung eines Antwortzeitverhaltens sollten Ausreißer ebenfalls berücksichtigt werden, auch wenn sie keinen direkten Einfluss auf den Kurvenverlauf haben.

Damit kann ein beobachtetes oder ein gewünschtes Antwortzeitverhalten nun sowohl mit der gewählten Wahrscheinlichkeitsverteilung simuliert als auch mit den fünf definierten Parametern aussagekräftig beschrieben werden.

8.5 Metadaten zur Beschreibung eines Literaturdienstes

Nachdem sowohl die strukturellen Eigenschaften als auch das Antwortzeitverhalten der ausgewählten Literaturdienste untersucht wurden, soll in diesem Abschnitt für die Beschreibung eines Dienstes ein Metadatensatz definiert werden, welcher die Informationen enthält, die für die Anfragebearbeitung eines Meta-Recherchesystems von Bedeutung sind. Die Struktur der Metadaten soll wie das Domänenmodell auch in Form einer DTD spezifiziert werden.

Jeder Literaturdienst wird durch ein eindeutiges Dienstkürzel beschrieben, über welches die Anfragebearbeitung den zum Literaturdienst gehörenden Wandler anspricht. Der Wandler kennt die URL des Literaturdienstes, kann die gegebenen Anfragen in das native Format des Literaturdienstes übersetzen und die empfangenen Ergebnisse in einem einheitlichen Format gemäß des intern verwendeten Domänenmodells an die Anfragebearbeitung zurückgeben.

Für die Anfragebearbeitung eines Meta-Recherchesystems sind die strukturellen Eigenschaften, das Antwortzeitverhalten und die Lastvorgaben eines Literaturdienstes von Interesse.

Zu den strukturellen Eigenschaften eines Dienstes zählen die Attribute, die bei der Formulierung von Anfragen verwendet werden können, und die Attribute, die vom Dienst

überhaupt geliefert werden. Dabei ist es wichtig zu wissen, auf welcher Informationsebene die Attribute zur Verfügung gestellt werden (vertikale Informationsportionierung). Weiterhin ist die horizontale Informationsportionierung von Bedeutung, d.h. wie viele Kurztitel jeweils auf einer Ergebnisseite präsentiert werden und in welcher Reihenfolge diese Kurztitel angeordnet werden können.

Zur Beschreibung des Antwortzeitverhaltens eines Literaturdienstes können mehrere Antwortzeitverteilungen angegeben werden. Eine Antwortzeitverteilung bezieht sich ja auf eine Anfrage bzw. auf eine Art von Anfrage, beispielsweise auf eine kleine oder große Anfrage oder auf eine Link-Expansion der Ebene 2 oder 3. Weiterhin kann die Ausreißerquote für einen Literaturdienst angegeben werden, diese ist i.d.R. allerdings unabhängig von den gestellten Anfragen bzw. Arten der Anfragen.

Webserver verwalten eine Reihe von parallelen Verbindungen. Lastvorgaben von Literaturdiensten beziehen sich üblicherweise auf die Begrenzung der parallelen Verbindungen, die von einem Meta-Recherchesystem gleichzeitig genutzt werden dürfen. Ein Literaturdienst kann mehrere Lastvorgaben machen, beispielsweise können einem Meta-Recherchesystem nachts mehr parallele Verbindungen zugestanden werden als tagsüber.

Die folgende DTD spezifiziert und konkretisiert die geschilderten Sachverhalte:

```
<!ELEMENT Dienst ( StrukturInfo? , AntwortzeitInfo? , LastInfo* )>
 <!ATTLIST Dienst Dienstkürzel CDATA #IMPLIED>

  <!ELEMENT StrukturInfo ( AnfragbAttr* , LieferbAttr* , SortierbAttr* )>
   <!ATTLIST StrukturInfo KurztitelStandard CDATA #IMPLIED>
   <!ATTLIST StrukturInfo KurztitelMaximal CDATA #IMPLIED>

    <!ELEMENT AnfragbAttr ( Attributname )>
    <!ELEMENT LieferbAttr ( Attributname )>
     <!ATTLIST LieferbAttr EbenenNummer CDATA #IMPLIED>
    <!ELEMENT SortierbAttr ( Attributname )>
     <!ATTLIST SortierbAttr Reihenfolge ( auf | ab ) #IMPLIED>
      <!ELEMENT Attributname (#PCDATA)>

  <!ELEMENT AntwortzeitInfo ( Ausreißerquote? , ZVerteilung* )>

    <!ELEMENT Ausreißerquote (#PCDATA)>
    <!ELEMENT ZVerteilung ( MinZeit , MaxZeit , SpitzBreite , SpitzAnteil)>
     <!ATTLIST ZVerteilung Art ( Anfr | Eb1 | Eb2 | ... | EbN ) #IMPLIED>
      <!ELEMENT MinZeit (#PCDATA)>
      <!ELEMENT MaxZeit (#PCDATA)>
      <!ELEMENT SpitzBreite (#PCDATA)>
      <!ELEMENT SpitzAnteil (#PCDATA)>

  <!ELEMENT LastInfo ( ParalleleVerbindungen, ZeitpunktVon, ZeitpunktBis )>

   <!ELEMENT ParalleleVerbindungen (#PCDATA)>
   <!ELEMENT ZeitpunktVon (#PCDATA)>
   <!ELEMENT ZeitpunktBis (#PCDATA)>
```

Bei der horizontalen Informationsportionierung wird sowohl die Standardeinstellung für die Größe der Kurztitelportionen als auch die dafür maximal mögliche Einstellung angegeben. Für die Attribute, die zur Sortierung der Kurztitelliste verwendet werden können, muss jeweils auch die Sortierreihenfolge angegeben werden. Damit können Sortierreihenfolgen, die sich nicht auf ein Attribut des Domänenmodells zurückführen lassen, nicht dargestellt werden.

Für die Beschreibung einer Antwortzeitverteilung werden die in Abschnitt 8.4 definierten „aussagekräftigen“ Parameter verwendet. Dadurch lässt sich von der Anfragebearbeitung leicht der Rahmen abstecken, in welchem die zu erwartenden Antwortzeiten liegen.

8.6 Resümee

In diesem Kapitel wurden die Grundlagen für die Konzeption und Umsetzung einer geeigneten Anfragebearbeitung im Rahmen eines Meta-Recherchesystems erarbeitet.

Dazu wurden zunächst zehn repräsentativ ausgewählte Literaturdienste auf ihre strukturellen Eigenschaften hin untersucht. Dabei hat sich gezeigt, dass die horizontalen Informationsportionierungen zum Teil so beeinflusst werden können, dass größere Ergebnisportionen geliefert werden können. Die vertikalen Informationsportionierungen lassen sich dagegen wenig beeinflussen. Mehr als 3 Ebenen zur Informationsaufteilung werden nur selten verwendet. Die Klassen der Bibliotheksdienste und der Buchhändler sind in der verfügbaren Attributmenge und in der Attributaufteilung recht unterschiedlich, können sich daher allerdings auch gut ergänzen.

Anschließend wurde das Antwortzeitverhalten der ausgewählten Literaturdienste untersucht. Im ersten Experiment wurden bei den meisten Diensten durchschnittliche Antwortzeiten zwischen 1 und 5 Sekunden identifiziert. Dabei wurden unterschiedliche Größen und Arten von Anfragen berücksichtigt. Die Einflussmöglichkeiten der Anfragebearbeitung beschränken sich dabei im Wesentlichen auf die Bestimmung der Länge der präsentierten Kurztitelliste und der Sortierreihenfolge. Durch längere Kurztitellisten ergeben sich verhältnismäßige Zeitvorteile. Bei der Wahl der Sortierreihenfolge konnten keine Zeitvorteile nachgewiesen werden, hier sollte die Anfragebearbeitung Sortierreihenfolgen wählen, die möglicherweise in anderen Bereichen Vorteile bringen, beispielsweise bei der Duplikaterkennung.

Im zweiten Experiment wurde eine typische Verteilung von Antwortzeiten gefunden, die zum einen hohe Standardabweichungen verursacht und zum anderen durch die Funktionsweise von Webservern auch plausibel erklärt werden kann. Daher wird zur Beschreibung des Antwortzeitverhaltens kein Durchschnittswert verwendet. Für die Simulation eines typischen Antwortzeitverhaltens kann die Kombination einer Weibull- und einer Gleichverteilung verwendet werden, für eine menschenlesbare Beschreibung des Kurvenverlaufs wurden vier aussagekräftige Parameter definiert.

Abschließend wurde ein Metadatenformat für die Beschreibung von Literaturdiensten spezifiziert. Dadurch stehen der Anfragebearbeitung Informationen über die eingebundenen Dienste zur Verfügung, die sie bei der Anfrageplanung berücksichtigen muss (z.B. anfragbare Attribute, lieferbare Attribute und vertikale Informationsportionierungen, Lastvorgaben) bzw. kann (z.B. sortierbare Attribute, horizontale Informationsportionierungen).

9 Konzeption und Umsetzung der Anfragebearbeitung

9.1 Überblick

Nachdem das vorangegangene Kapitel die Grundlagen für die Anfragebearbeitung untersucht hat, beschäftigt sich dieses und das folgende Kapitel nun mit der Konzeption und Umsetzung der Anfragebearbeitung. Aufgabe der Anfragebearbeitung eines Meta-Recherchesystems ist es, aus einer gegebenen Anfrage (in der vorliegenden Arbeit also aus einer gegebenen MLS-QL-Anfrage) die notwendigen Aufrufe an die Wandler abzuleiten, diese Aufrufe zu bearbeiten, die Einzelergebnisse zu empfangen und schließlich aus den Einzelergebnissen eine integrierte Ergebnismenge aufzubauen. Für den Vorgang der Anfragebearbeitung sind eine Reihe von Schritten notwendig. Die in Kapitel 4 vorgestellte Grobarchitektur des Benutzerassistenten (Abbildung 4.1) sieht dazu drei Komponenten vor, eine Komponente zur Anfrageplanung, eine Komponente zur Anfrageausführung und eine Komponente zur Ergebnisintegration.

In diesem Kapitel werden zunächst noch einmal die Anforderungen zusammengestellt, die die Anfragebearbeitung betreffen. Daraufhin wird die Struktur des vorgeschlagenen Lösungsansatzes vorgestellt und die Funktionsaufteilung und der Funktionsumfang der drei Hauptkomponenten detaillierter erläutert (Abschnitt 9.2). Anschließend wird die Grundidee der Handlungsspielräume für die Umsetzung des Lösungsansatzes formuliert und die weitere Vorgehensweise daraus abgeleitet.

In Abschnitt 9.3 werden bei jeder der drei Komponenten die Handlungsspielräume identifiziert. In den darauffolgenden Abschnitten werden dann alternative Verfahren und Algorithmen für die Besetzung dieser Handlungsspielräume erarbeitet. Für die Komponente der Anfrageplanung geschieht das in Abschnitt 9.4, für die Komponente der Anfrageausführung in Abschnitt 9.5. Die Handlungsalternativen für die Komponente der Ergebnisintegration werden in Kapitel 10 erarbeitet.

Um die Auswirkungen der unterschiedlichen Strategien auf das Antwortzeitverhalten der Anfragebearbeitung beurteilen zu können, werden Bewertungsmaße benötigt, die die Schnelligkeit der Ergebnislieferung beobachten. Anhand der Bewertungsmaße sollen die Strategien ermittelt werden, die dem Benutzer die Ergebnisse am schnellsten bzw. die bevorzugten Ergebnisse möglichst schnell liefern können. Mit der Entwicklung von geeigneten Bewertungsmaßen beschließt Kapitel 11 den zweiten Teil der Arbeit, der sich mit der Konzeption und Umsetzung einer geeigneten Anfragebearbeitung beschäftigt.

9.2 Anforderungen und Lösungsansatz

Zunächst sollen noch einmal die Anforderungen, die den Bereich der Anfragebearbeitung betreffen, zusammengestellt werden. A1 wird aufgrund der Integrationsforderung hinzugenommen, da die Integration, d.h. die Zusammenführung der horizontalen und vertikalen Informationsportionen, im Rahmen der Anfragebearbeitung erfüllt wird. Die Benutzungsschnittstelle präsentiert lediglich eine bereits integriert vorliegende Ergebnismenge.

- A1: Einheitliche, integrierte Benutzungsschnittstelle
- A5: Vollständige Anfrageplanung und -bearbeitung
- A6: Duplikatbehandlung
- A7: Schnelle Antwortzeiten
- A8: Berücksichtigung von Lastvorgaben

Zur Lösung der Anforderungen wird in der vorliegenden Arbeit ein Ansatz vorgeschlagen, der den Vorgang der Anfragebearbeitung in drei Phasen aufteilt: Zunächst wird ein Anfrageplan für die Bearbeitung der Benutzeranfrage erstellt. Dieser Anfrageplan wird anschließend auf der Basis der vorliegenden Wandler bzw. Literaturdienste ausgeführt und abgearbeitet. Dann werden die einzelnen Teilergebnisse aufgesammelt und in einer Ergebnismenge integriert. Diese drei Phasen der Anfragebearbeitung spiegeln sich in der Grobarchitektur des Benutzerassistenten wider (vgl. Abbildung 4.1), in der Komponente der Anfrageplanung, der Anfrageausführung und in der Komponente der Ergebnisintegration. Jede Komponente beinhaltet die Funktionen, Verfahren und Algorithmen, die zur Bearbeitung der Anfrage in dieser Phase notwendig sind. Abbildung 9.1 zeigt eine detailliertere Darstellung des Ausschnitts des Benutzerassistenten, der die Anfragebearbeitung betrifft.

Anfrageplanung. Die Komponente der Anfrageplanung erhält als Eingabe eine vollständige MLS-QL-Anfrage. Dazu wird ein Anfrageplan erstellt, der aus den betreffenden Literaturdiensten alle gewünschten und verfügbaren Informationen beschafft (A5). Die Anfrageplanung sieht dabei generell vor, dass die betreffenden Literaturdienste in paralleler Weise befragt werden. Ein Teilanfrageplan, der sich auf einen speziellen Literaturdienst bezieht, besteht aus einer oder auch aus mehreren Anfragen mit entsprechenden Parametern und aus Angaben, welche Informationsebenen zu einer Anfrage berücksichtigt werden müssen, um die geforderten Attribute des Benutzers zu beschaffen. Zur Erstellung der parallelen Teilanfragepläne werden die Metadaten der Literaturdienste benötigt. Die Teilanfragepläne dürfen nur solche Bedingungen enthalten, die vom jeweiligen Literaturdienst auch unterstützt werden (anfragbare Attribute). Über die Anfrageparameter können Sortierungen und horizontale Informationsportionen gefordert werden, die von dem Literaturdienst angeboten werden.

Anfragebedingungen, die sich auf Attribute beziehen, die bei einem Literaturdienst nicht angefragt werden können, werden im Rahmen der Ergebnisintegration durch entsprechende

Filterbedingungen nachgebildet. Die benötigten Filterbedingungen werden zusammen mit den anderen Steuerinformationen (Lastvorgaben, Präferenzen, Duplikatbedingung, Gruppier- und Sortiervorgabe) an die folgenden Komponenten weitergeleitet. Die Lastvorgaben sind direkt für die Anfrageausführung bestimmt, die anderen Angaben müssen erst bei der Ergebnisintegration berücksichtigt werden.

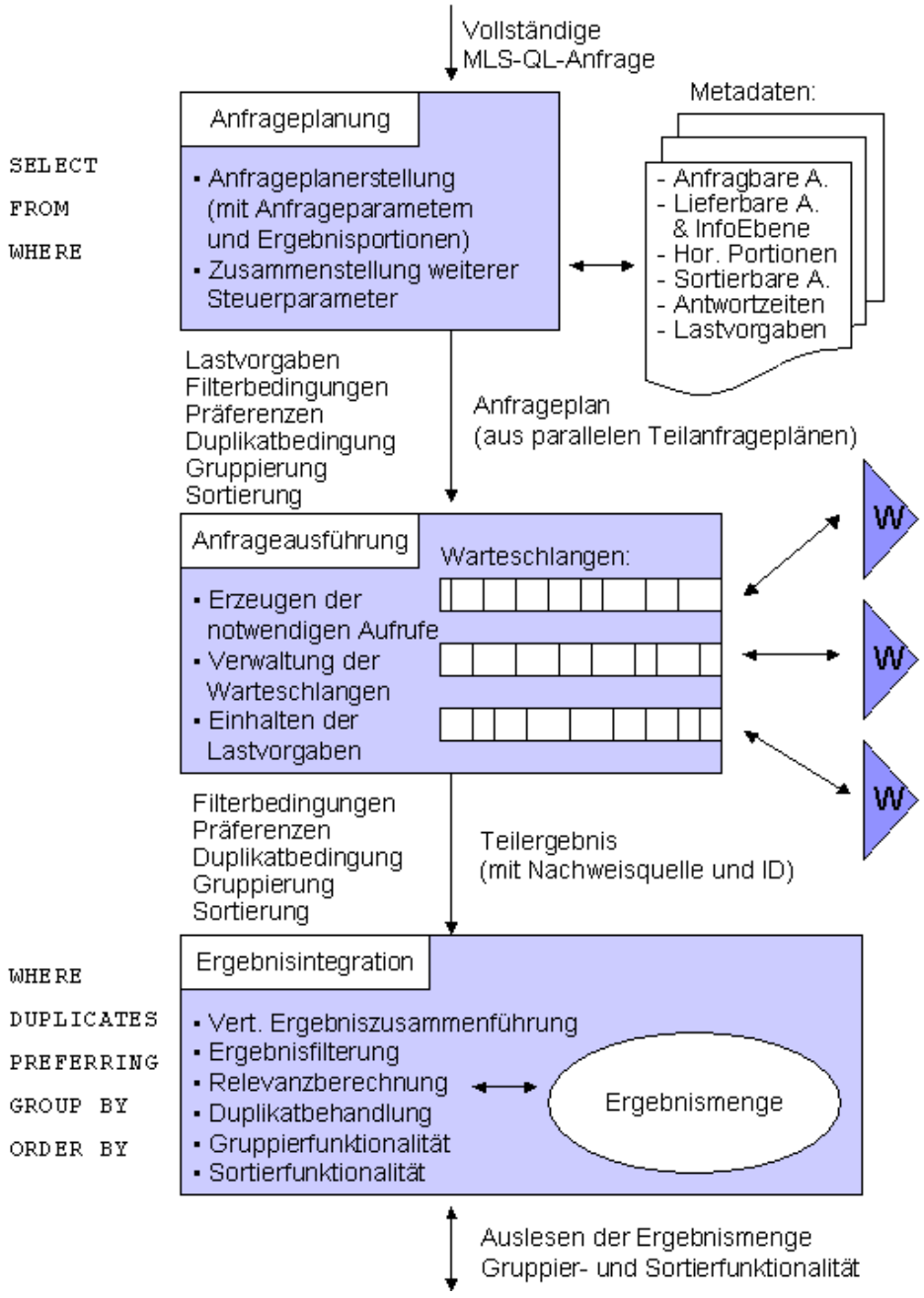


Abbildung 9.1: Architektur der Anfragebearbeitung

Anfrageausführung. Die Komponente der Anfrageausführung erhält den generierten Anfrageplan, der abgearbeitet werden muss. Dazu sind die Anfragen mit den angegebenen Bedingungen und Parametern an die entsprechenden Wandler abzusetzen und die gefundenen Ergebnisse jeweils bis zur angegebenen Informationsebene zu verfolgen (A5). Die dazu notwendigen Aktionen sind Aufrufe der Wandler, die im Wesentlichen den zwei Arten von HTTP-Aufrufen (Anfragen und Link-Expansionen, vgl. Abschnitt 8.3.1) entsprechen, jeweils allerdings in einem einheitlichen Format abgesetzt werden können. Die Anfrageausführung baut für jeden Literaturdienst eine Warteschlange mit den notwendigen Aktionen auf. Zu Beginn enthält eine Warteschlange nur eine oder mehrere Anfragen. Nachdem die Anzahl der gefundenen Ergebnisse ermittelt wurde, können die notwendigen Aktionen, die den Link-Expansionen für tiefere Informationsebenen entsprechen, in die Warteschlange eingefügt werden. Die Einhaltung der Lastvorgaben wird von der Komponente der Anfrageausführung durchgesetzt. Von jeder Warteschlange kann nur eine bestimmte Zahl von Aktionen gleichzeitig bearbeitet werden (A8). Die eintreffenden Teilergebnisse werden sofort an die Komponente der Ergebnisintegration weitergegeben (A7). Dabei enthält jedes Teilergebnis die Angabe des Dienstes, bei dem es gefunden wurde, und die Position in dessen Ergebnisliste. Dadurch können die Teilergebnisse bei der Ergebnisintegration zusammengeführt werden.

Ergebnisintegration. Die Komponente der Ergebnisintegration baut aus den kontinuierlich eintreffenden einheitlichen Teilergebnissen eine integrierte Ergebnismenge auf (A1). Im Rahmen der Ergebnisintegration können fehlende Funktionalitäten der Literaturdienste ausgeglichen werden. Die gelieferten Ergebnisse werden so gefiltert, dass nur korrekte, d.h. den Filterbedingungen entsprechende Ergebnisse in die Ergebnismenge aufgenommen werden. Für jedes eintreffende Teilergebnis wird der Relevanzwert bezüglich der gegebenen Präferenzen ermittelt. Beim Verschmelzen der Teilergebnisse addieren sich die jeweiligen Relevanzwerte. Beim Einfügen eines neuen Ergebnisses in die Ergebnismenge werden bereits vorhandene Duplikate ermittelt (A6). Die identifizierten Duplikate werden auf der Basis der XML-Datensätze miteinander verschmolzen.

Die Ergebnismenge hat die Struktur einer geschachtelten Liste von XML-Datensätzen. Die erste Ebene der Liste spiegelt die Sortierreihenfolge wider, die zweite Ebene wird für Gruppierungen verwendet. Die Gruppier- und Sortieroperationen arbeiten auf dieser Listenstruktur. Die Operatoren bauen die in der Anfrage geforderte Listenstruktur der Ergebnisse auf, können aber auch über die an der Benutzungsschnittstelle angebotenen interaktiven Nachbearbeitungsfunktionen aufgerufen werden. Die Strukturierungsoperatoren und die Duplikatbehandlung werden mit nicht blockierenden Algorithmen realisiert, d.h. es entstehen keine zusätzlichen längeren Wartezeiten im Verlauf der Anfragebearbeitung (A7).

Abbildung 9.1 stellt am linken Rand der Komponenten dar, welche MLS-QL-Konstrukte der gegebenen Anfrage in der jeweiligen Komponente verarbeitet werden. Beim Informationsfluss durch die Komponenten wird zwischen den Anfragedaten (rechts neben den senkrechten Pfeilen) und den Steuerinformationen (links neben den senkrechten Pfeilen) unterschieden.

Grundidee und weitere Vorgehensweise. Die größte Herausforderung der Anfragebearbeitung besteht in der möglichst schnellen Bereitstellung der Ergebnisse. Diese Anforderung wird prinzipiell durch den kontinuierlichen Ergebnismengenaufbau und den Einsatz von nicht blockierenden Operatoren zur Duplikaterkennung, Gruppierung und Sortierung unterstützt. Die Komponente der Ergebnispräsentation kann die Ergebnisse verarbeiten bzw. sie dem Benutzer anzeigen, sobald sie in der integrierten Ergebnismenge vorliegen. Durch den kontinuierlichen Ergebnismengenaufbau wird die initiale Wartezeit für den Benutzer deutlich verkürzt. Dadurch empfindet er die Ergebnislieferung als schneller, obwohl die Zeit, bis die kontinuierliche Ergebnismenge vollständig aufgebaut ist und bis eine mit blockierenden Operatoren realisierte Ergebnismenge präsentiert werden könnte, in etwa dieselbe ist.

Bei der möglichst schnellen Bereitstellung der Ergebnisse ist zu beachten, dass dabei auch die subjektive Empfindung des Benutzers eine Rolle spielt, beispielsweise im Hinblick auf die Lieferung von relevanten Ergebnissen bzw. Informationen. Werden die relevanten Ergebnisse bzw. Informationen schneller geliefert, so entsteht auch insgesamt der Eindruck, dass die Ergebnisse schneller verfügbar sind.

Daraus leitet sich die Idee des vorgeschlagenen Lösungsansatzes ab: Jede der drei Komponenten zur Anfragebearbeitung weist gewisse Handlungsspielräume bei der Realisierung ihrer Funktionalitäten auf. Zunächst sollen für jede Komponente diese Handlungsspielräume identifiziert werden. Daraufhin sollen die Handlungsspielräume mit unterschiedlichen Verfahren bzw. Algorithmen besetzt werden. Durch die Alternativen sollen die Ergebnisse entweder insgesamt schneller oder bevorzugte bzw. relevante Ergebnisse schneller geliefert werden. Bei der Erarbeitung der Strategien müssen die Auswirkungen auf das Antwortzeitverhalten noch nicht im Detail analysiert werden. Dazu erfolgen im weiteren Verlauf der Arbeit ausführliche praktische Untersuchungen.

Anschließend sollen Maße für die Charakterisierung und die Bewertung des Antwortzeitverhaltens der Anfragebearbeitung entwickelt werden. Diese Maße sollen nicht nur den Ergebnisdurchsatz (d.h. die gelieferten Ergebnisse pro Zeit) ermitteln, sondern auch die subjektiven Empfindungen der Benutzer bzgl. schneller Ergebnisse widerspiegeln. Der Fokus dieser Maße liegt auf dem kontinuierlichen Aufbau der Ergebnismenge. Die Bewertungsmaße sollen dazu dienen, die Auswirkungen der entwickelten Handlungsvarianten ermitteln und miteinander vergleichen zu können und danach die „besten“ Varianten bestimmen zu können.

Die weitere Vorgehensweise gestaltet sich folgt: Im nächsten Abschnitt werden die Handlungsspielräume der drei Komponenten erarbeitet (Abschnitt 9.3). Anschließend werden für jede Komponente unterschiedliche Strategien für die Besetzung der Handlungsspielräume entwickelt. Für die Komponente der Anfrageplanung geschieht das in Abschnitt 9.4, für die Komponente der Anfrageausführung in Abschnitt 9.5. Die Handlungsalternativen für die Komponente der Ergebnisintegration werden in Kapitel 10 erarbeitet. Die Entwicklung

geeigneter Bewertungsmaße erfolgt in Kapitel 11. Praktische Untersuchungen zur Validierung der entwickelten Strategien werden im dritten Teil der Arbeit durchgeführt.

9.3 Handlungsspielräume bei der Anfragebearbeitung

Anfrageplanung. Bei der Erstellung eines korrekten Anfrageplans zu einer gegebenen MLS-QL-Anfrage bestehen bei den meisten Schritten keine Handlungsspielräume: Anhand der Metadaten muss ermittelt werden, ob die Anfragebedingungen von den Literaturdiensten bearbeitet werden können oder ob zur Bearbeitung Filterbedingungen bei der Ergebnisintegration benötigt werden. Die Ebenen der gewünschten Attribute müssen ermittelt werden. Durch die gegebene MLS-QL-Anfrage und die Metadaten sind die Steuerinformationen bereits eindeutig bestimmt.

Der Handlungsspielraum bei der Anfrageplanung besteht im Wesentlichen darin, die gegebene MLS-QL-Anfrage nicht notwendigerweise als eine einzige Anfrage an die Wandler weiterzuleiten, sondern daraus ggf. mehrere Teilanfragen zu bilden, die in ihrer Gesamtheit allerdings wieder der Semantik der gegebenen Anfrage entsprechen müssen. Wenn beispielsweise Bücher zum Thema Java gesucht werden sollen, so können zunächst die deutschsprachigen Bücher, dann die englischsprachigen Bücher und schließlich die Bücher in anderen Sprachen gesucht werden, sofern der betreffende Dienst Anfrageformulierungen mit Bedingungen an das Sprach-Attribut zulässt. Durch diese Vorgehensweise ist es sicherlich nicht möglich, die Ergebnisse schneller zu ermitteln, da die Gesamtzahl der gefundenen Treffer bei beiden Varianten gleich ist und damit eben auch die Zahl (und Dauer) an benötigten Link-Expansionen gleich ist. Allerdings können durch geschickte *Anfragezerlegungen* relevantere Ergebnisse zu einem früheren Zeitpunkt geliefert werden.

Weitere Handlungsspielräume der Anfrageplanung bestehen in den Parameterbelegungen der Anfragen, nämlich in der gewählten Sortierreihenfolge, die von den Ergebnissen des Literaturdienstes gefordert werden kann, und in der gewählten Größe der horizontalen Ergebnisportionen. In Kapitel 8 konnten keine Aussagen darüber getroffen werden, ob bestimmte Sortierreihenfolgen von den Literaturdiensten schneller geliefert werden können als andere. Bei der Auswahl der Sortierreihenfolge ist es zum einen denkbar, sich an der gewünschten Sortierreihenfolge der gegebenen Anfrage zu orientieren, um dadurch den Aufwand für Umsortierungen bei der Ergebnisintegration gering zu halten. Zum anderen ist es aber auch denkbar, dass sich eine Sortierreihenfolge, die sich an den Attributen orientiert, die zur Duplikaterkennung verwendet werden, vorteilhaft auf die Bearbeitungszeit bei der Duplikaterkennung auswirken kann. Dazu sollte allerdings auch von möglichst allen befragten Diensten die gewünschte Sortierreihenfolge angeboten werden. Bei den Experimenten zum Antwortzeitverhalten hat sich herausgestellt, dass größere horizontale Ergebnisportionen ein günstigeres Verhältnis zwischen Ergebnisanzahl und Antwortzeit

aufweisen. Insofern ist es im Allgemeinen sinnvoll, von jedem Literaturdienst die größtmöglichen horizontalen Ergebnisportionen abzurufen.

Anfrageausführung. Beim Erzeugen der notwendigen Link-Expansions-Aufrufe und beim Einhalten der Lastvorgaben bestehen bei der Anfrageausführung keine Handlungsspielräume. Der Handlungsspielraum liegt bei der Anfrageausführung lediglich in der *Reihenfolge, in der die Aktionen in den Warteschlangen abgearbeitet werden*. Müssen beispielsweise zu jedem Dokument Informationen der zweiten und der dritten Ebene beschafft werden, so sind mehrere Vorgehensweisen denkbar: Entweder es können zunächst von allen Dokumenten die Informationen der zweiten Ebene angefordert werden und anschließend wieder von allen Dokumenten die Informationen der dritten Ebene. Oder es werden von einem Dokument zuerst die Informationen der zweiten und dritten Ebene besorgt, bevor das nächste Dokument bearbeitet wird. Diese Vorgehensweisen würden sozusagen einer Breiten- bzw. einer Tiefensuche entsprechen, die allerdings jeweils mit dem entsprechenden Parallelitätsgrad ausgeführt werden sollte. Weiterhin sind natürlich auch beliebige Mischformen der Breiten- und der Tiefensuche denkbar. Durch verschiedene Besetzungen dieses Handlungsspielraums können die Ergebnisse zwar insgesamt nicht schneller geliefert werden, aber es können möglicherweise die relevanteren Informationen schneller geliefert werden.

Ergebnisintegration. Soll die Ergebnismenge kontinuierlich aufgebaut werden, bestehen nahezu keine Handlungsspielräume: Die Ergebniszusammenführung kann aufgrund einer eindeutigen Kennung aus ID und Literaturdienst gezielt erfolgen, ohne dass die bisherigen Ergebnisse in der Ergebnismenge aufgegriffen werden müssen. Die Ergebnisfilterung hält die eintreffenden Teilergebnisse solange zurück, bis aufgrund der vorhandenen Informationen entschieden werden kann, ob das Ergebnis zur Ergebnismenge gehört oder nicht. Die Relevanzberechnungen erfolgen direkt beim Einfügen eines Ergebnisses oder beim Hinzufügen eines neuen Teilergebnisses in die Ergebnismenge. Die Gruppier- und Sortierfunktionen sind in ihrem minimalen Aufwand bekannt (O(n log n)).

Die Duplikatbehandlung ist sicherlich die aufwendigste Funktion im Rahmen der Ergebnisintegration. Durch die MLS-QL-Anfrage werden die gewünschten Attribute mit den entsprechenden Gleichheitsoperatoren vorgegeben. Als Vergleichsoperatoren sollen sowohl exakte als auch tolerante Vergleiche angeboten werden. Exakte Vergleichsoperatoren sind für Vergleiche von Zeichenketten nicht ausreichend, da sich im Bereich der bibliographischen Informationen leicht Tippfehler einschleichen oder Texte uneinheitlich abgekürzt werden. Tolerante Vergleiche von Zeichenketten sind allerdings deutlich aufwendiger zu realisieren als exakte Vergleiche. Der Handlungsspielraum bei der Ergebnisintegration besteht also im Wesentlichen darin, *unterschiedliche Algorithmen für tolerante Zeichenkettenvergleiche* zur Verfügung zu stellen. Diese Algorithmen sollen dann sowohl im Hinblick auf ihre Qualität beim Identifizieren von Duplikaten als auch im Hinblick auf ihren Aufwand überprüft werden. Dadurch soll eine Duplikaterkennung den kontinuierlichen Aufbau der Ergebnismenge möglichst wenig verzögern.

Die folgenden Abschnitte widmen sich nun jeweils einer detaillierteren Ausarbeitung der Handlungsalternativen für die identifizierten Gestaltungsfreiräume. Aufgrund der zentralen Problematik der Duplikaterkennung wird diesem Thema ein vollständiges Kapitel gewidmet.

9.4 Strategien zur Anfragezerlegung

Der wesentliche Handlungsspielraum bei der Anfrageplanung ist die Zerlegung einer gegebenen Anfrage in mehrere Teilanfragen.

Definition 9.1: Eine *Teilanfrage* zu einer gegebenen Anfrage ist eine Anfrage, die neben den Bedingungen der gegebenen Anfrage noch zusätzliche Bedingungen enthält und damit eine Untermenge der Ergebnisse liefert, die die gegebene Anfrage findet.

Definition 9.2: Eine Anfragezerlegung zu einer gegebenen Anfrage heißt *vollständig*, wenn die entstandenen Teilanfragen insgesamt genau dieselben Ergebnisse liefern wie die gegebene Anfrage.

Durch die Angabe von Präferenzen kann der Benutzer seine Vorliebe für bestimmte Eigenschaften von Dokumenten ausdrücken. Diese Vorlieben haben keinen Einfluss auf die Menge der gefundenen Ergebnisse. Sie geben aber Hinweise darauf, welche Ergebnisse für den Benutzer relevanter sind als andere. Die Technik der Anfragezerlegung kann dazu verwendet werden, um die Ergebnisse in einer gewissen Reihenfolge zu beschaffen, sofern die bei der Zerlegung entstandenen Teilanfragen nacheinander ausgeführt werden. Werden bei der Anfragezerlegung die Präferenzen des Benutzers berücksichtigt, können die relevanteren Ergebnisse zuerst beschafft und angezeigt werden.

Gegeben sei eine Anfrage mit Bedingung B und mit n gewichteten Präferenzen $w_i * P_i$, wobei $1 \leq i \leq n$. Eine Präferenz P_i wird als Bedingung aufgefasst, die wahr ist, wenn alle Ergebnisse die bevorzugten Eigenschaften aufweisen.

Definition 9.3: Eine Anfragezerlegung heißt *präferenzbasiert*, wenn jede der 2^i Teilanfragen T_j aus der Bedingung B und einer unterschiedlichen Klausel K_j der kanonischen Disjunktiven Normalform (kDNF; $1 \leq j \leq 2^i$) über der Menge der Präferenzen P_i besteht, d.h. $T_j = B \wedge K_j$.

Satz 9.4: Jede präferenzbasierte Anfragezerlegung ist vollständig.

Beweis: Es muss also gezeigt werden, dass die durch eine präferenzbasierte Anfragezerlegung entstehenden Teilanfragen genau dieselben Ergebnisse liefern wie die gegebene Anfrage. Jedes von der Anfragezerlegung gefundene Ergebnis erfüllt genau eine Teilanfrage T_j ($1 \leq j \leq 2^i = m$). Jede Teilanfrage kann nun durch die Bedingung und die entsprechende Klausel der kDNF ersetzt werden. Anschließend kann der Ausdruck durch das Distributivgesetzt umgeformt werden. Die disjunktive Verknüpfung der Klauseln entspricht nun genau der

kDNF, diese hat den Wahrheitswert *wahr*. Damit erfüllt jedes von der Anfragezerlegung gefundene Ergebnis auch genau die gegebene Anfragebedingung B.

$$\begin{aligned} T_1 \vee \ldots \vee T_m &= (B \wedge K_1) \vee \ldots \vee (B \wedge K_m) \\ &= B \wedge (K_1 \vee \ldots \vee K_m) \\ &= B \wedge \textit{wahr} \\ &= B \end{aligned}$$

Eine geeignete Reihenfolge der Teilanfragen einer präferenzbasierten Anfragezerlegung kann so bestimmt werden, dass die Zahl der positiv in den Klauseln erscheinenden Präferenzen kontinuierlich abnimmt. D.h. die erste Teilanfrage enthält nur positiv formulierte Präferenzen, die folgenden Teilanfragen enthalten jeweils eine negativ formulierte Präferenz und die darauf folgenden Teilanfragen enthalten jeweils zwei negativ formulierte Präferenzen usw. Dabei werden zuerst die Präferenzen P_i negiert, die die niedrigste Gewichtung w_i aufweisen.

Beispiel: Gegeben sei eine Anfrage mit Bedingung B und drei gewichteten Präferenzen $w_i * P_i$: $1{,}0 * P_1$; $0{,}5 * P_2$; $0{,}7 * P_3$. Eine präferenzbasierte Anfragezerlegung würde demnach folgende Reihenfolge für die Teilanfragen wählen:

$$\begin{aligned} T_1 &= B \wedge P_1 \wedge P_2 \wedge P_3 \\ T_2 &= B \wedge P_1 \wedge \neg P_2 \wedge P_3 \\ T_3 &= B \wedge P_1 \wedge P_2 \wedge \neg P_3 \\ T_4 &= B \wedge \neg P_1 \wedge P_2 \wedge P_3 \\ T_5 &= B \wedge P_1 \wedge \neg P_2 \wedge \neg P_3 \\ T_6 &= B \wedge \neg P_1 \wedge \neg P_2 \wedge P_3 \\ T_7 &= B \wedge \neg P_1 \wedge P_2 \wedge \neg P_3 \\ T_8 &= B \wedge \neg P_1 \wedge \neg P_2 \wedge \neg P_3 \end{aligned}$$

Zum Abschluss wollen wir die präferenzbasierte Anfragezerlegung anhand eines Anwendungsbeispiels illustrieren. Dazu betrachten wir eine präferenzbasierte Anfragezerlegung für eine Anfrage nach Büchern über Java, die die Präferenzen `1,0 * möglichst_aktuell(Jahr)` und `1,0 * lieber_deutsch(Sprache)` enthält (vgl. Beispiel S. 172). Die Bedingungen der entstehenden Teilanfragen sehen folgendermaßen aus:

```
T1:     WHERE Originaltitel LIKE "%Java%"
        AND Jahr >= 1999
        AND Sprache = "de"

T2:     WHERE Originaltitel LIKE "%Java%"
        AND Jahr < 1999
        AND Sprache = "de"
```

```
T3:     WHERE Originaltitel LIKE "%Java%"
        AND Jahr >= 1999
        AND Sprache != "de"

T4:     WHERE Originaltitel LIKE "%Java%"
        AND Jahr < 1999
        AND Sprache != "de"
```

Zunächst werden die Bücher gesucht, die beide Präferenzen erfüllen (T_1). Als nächstes werden die Bücher beschafft, die eine der beiden Präferenzen erfüllen. Da in der Beispielanfrage beide Präferenzen dieselbe Gewichtung aufweisen, kann die Reihenfolge von T_2 und T_3 beliebig gewählt werden. Ansonsten würden die Dokumente zuerst beschafft werden, die die Präferenz mit der höheren Gewichtung erfüllen. In der letzten Teilanfrage (T_4) werden die übrigen Bücher beschafft, die keine der beiden Präferenzen erfüllen.

9.5 Strategien zur Reihenfolgebestimmung

Die Aufgabe der Anfrageausführung besteht in der Verwaltung der benötigten Warteschlangen und in der Einhaltung der vorgegebenen Lastbeschränkungen. Als Eingabe erhält die Komponente der Anfrageausführung einen Anfrageplan, der für jeden zu befragenden Literaturdienst die entsprechende Anfrage bzw. die entsprechenden Teilanfragen (s. Abschnitt 9.4) enthält. Jede Anfrage enthält dabei die Anfragebedingungen, die Anzahl der zu beschaffenden Informationsebenen, die Größe der horizontalen Informationsportionen und die gewünschte Sortierreihenfolge.

Zu einer gegebenen Anfrage muss die Anfrageausführung nun die notwendigen Aktionen herleiten und ausführen. Eine Aktion entspricht dem Beschaffen einer Informationsportion. Zusammen mit der ersten Informationsportion wird i.d.R. die Gesamtzahl der vom Literaturdienst gefundenen Treffer übermittelt. Sobald die Anzahl der Treffer zu einer Anfrage bekannt sind, besteht Klarheit darüber, wie viele und welche Art von Informationsportionen zur Beantwortung der Anfrage beschafft werden müssen. Diese Informationen können allerdings nicht unabhängig voneinander beschafft werden, da die Informationsportionen jeweils Verweisinformationen (d.h. URLs) enthalten, die zum Aufruf der „nächsten" Informationsportion notwenig sind. Der Menge der Informationsportionen ist also eine gewisse Struktur aufgeprägt, die Bedingungen bzgl. der Aufrufreihenfolge vorgibt.

Abbildung 9.2 zeigt die Struktur der Informationsportionen. Die Portionen der ersten Ebene ($P_{1,j}$) stellen die Kurztitellisten dar, die zu einer Anfrage geliefert werden. Zur Ausführung einer Anfrage wird im ersten Schritt $P_{1,1}$ beschafft, um die Anzahl der gefundenen Treffer zu ermitteln. Daraus kann zusammen mit der gegebenen Größe der horizontalen Informationsportionen (hier: 10) berechnet werden, wie viele Portionen der ersten Ebene beschafft werden müssen. Die Abhängigkeiten zwischen den Portionen der ersten Ebene sind

insofern nicht zwingend und mit durchbrochenen Pfeilen dargestellt, als es prinzipiell auch möglich ist, Portionen sozusagen „auf gut Glück“ anzufordern. Dadurch können allerdings auch Fehlermeldungen auftreten, wenn zu einer Anfrage Trefferportionen angefordert werden, die dazu gar nicht existieren.

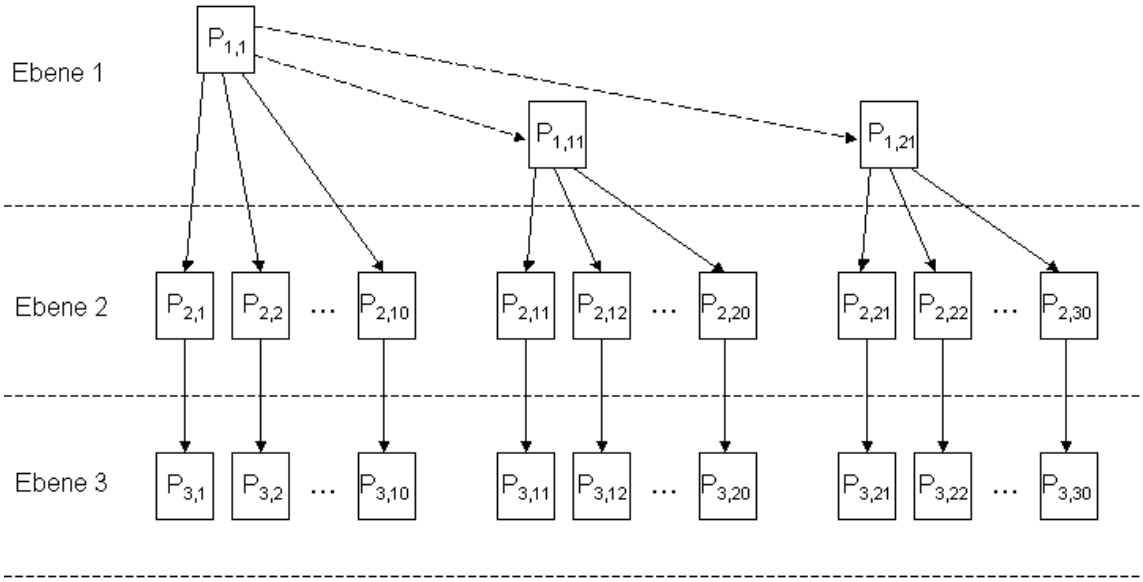


Abbildung 9.2: Aufruf-Abhängigkeiten zwischen Informationsportionen

Die durchgezogenen Pfeile zeigen die „echten“ Abhängigkeiten zwischen den Informationsportionen. Informationsportionen der zweiten Ebene können erst dann angefordert werden, wenn die entsprechende Kurztitelliste (mit den Verweisinformationen) bereits existiert. Analog dazu kann zu einem Dokument die Informationsportion der dritten Ebene erst abgerufen werden, wenn die Informationsportion der zweiten Ebene bereits vorhanden ist.

Darüber hinaus bestehen keine weiteren Beschränkungen bzgl. der Aufrufreihenfolge für die Anfrageausführung. Sobald eine Kurztitelliste vorhanden ist, können die entsprechenden Portionen der zweiten Ebene in beliebiger Reihenfolge angefordert werden. Ist eine Informationsportion der zweiten Ebene eingetroffen, kann entweder die entsprechende Portion der dritten Ebene oder eine weitere Portion der zweiten Ebene besorgt werden.

In Abschnitt 9.2 haben wir zur Beschreibung dieses Vorgangs das Modell der Warteschlangen eingeführt. In einer Warteschlangen befinden sich zu einem Zeitpunkt immer die Informationsportionen, die als nächste bezogen werden können, weil die benötigten Verweisinformationen bereits vorhanden sind (s. Abbildung 9.3). Zu Beginn einer Anfrageausführung (Zeitpunkt t1) enthält die Warteschlange i.d.R. nur $P_{1,1}$ (oder auch $P_{1,j}$, falls Fehlermeldungen in Kauf genommen werden). Sobald eine Informationsportion angefordert wird, wird die entsprechende Aktion (hier zuerst $P_{1,1}$) aus der Warteschlange entfernt. Ist das Ergebnis (mit

Verweisinformationen für weitere Aktionen) eingetroffen, werden diese nächsten möglichen Aktionen (hier $P_{2,1}$ bis $P_{2,10}$) in die Warteschlange eingefügt (Zeitpunkt t2). Nach der Ausführung von Aktion $P_{2,1}$ wird Aktion $P_{3,1}$ möglich (Zeitpunkt t3) usw.

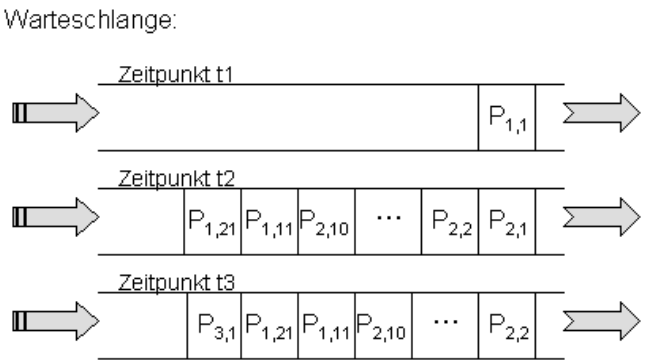


Abbildung 9.3: Warteschlange zur Koordinierung der Anfrage-Aktionen

Durch dieses Warteschlangenmodell ist es bei der Anfrageausführung in einfacher Weise möglich, einen bestimmten Parallelitätsgrad zu erreichen. Für jeden Literaturdienst wird genau eine Warteschlange verwaltet. Zu einem Zeitpunkt soll immer nur eine gegebene Anzahl an parallelen Aufrufen durchgeführt werden (Anforderung A8). Dazu kann zu jeder Warteschlange eine entsprechende Anzahl an Kontrollfäden erzeugt werden, die jeweils die erste Aktion der Warteschlange ausführen und anschließend die neuen möglichen Aktionen in die Warteschlange einfügen.

Der Handlungsspielraum bei der Anfrageausführung besteht in der Reihenfolge, in der die Menge der zur Beantwortung der Anfrage notwendigen Aktionen ausgeführt wird. Die Ausführungsreihenfolge der Aktionen kann dadurch beeinflusst werden, dass neue Aktionen nicht zwingend am Ende, sondern an eine bestimmte andere Position der Warteschlange eingefügt werden.

Das Verfahren, welches neue Aktionen stets am Ende der Warteschlange einfügt, wollen wir als *First Come First Serve (FCFS)* bezeichnen. Dabei entwickelt sich die Reihenfolge der Informationsbeschaffungen gemäß des Zeitpunkts, an dem die notwendigen Verweisinformationen für den Aufruf verfügbar waren.

Andere Varianten bzgl. der Reihenfolge der Informationsbeschaffungen lassen sich an der Baumstruktur von Abbildung 9.2 festmachen. Dabei sind Verfahren denkbar, die die Informationen gemäß einer Breitensuche oder einer Tiefensuche anfordern. In den Experimenten in Abschnitt 8.3 wurde deutlich, dass die Antwortzeiten von Informationsportionen verschiedener Stufen unterschiedlich sein können. Dabei waren z.B. die Antwortzeiten der Verfügbarkeitsinformationen (Stufe 3) deutlich länger als bei den bibliographischen Informationen (Stufe 2). In einem solchen Fall würde sich beispielsweise eine Breitensuche anbieten, die zunächst die Informationsportionen beschafft, die schneller zu beziehen sind. Eine

Tiefensuche wäre dagegen sinnvoll, wenn der Benutzer besonders an Attributen bzw. Eigenschaften von Dokumenten interessiert ist, die sich auf Stufe 3 oder tiefer befinden.

Im Rahmen des vorliegenden Lösungsansatzes kann eine Breitensuche bzw. eine Tiefensuche dadurch realisiert werden, dass die neuen Aktionen nicht am Ende, sondern an einer anderen Stelle der Warteschlange eingefügt werden. Sei $P_{i,j}$ eine neue Aktion.

Für die *Breitensuche* gilt: Die Warteschlange wird von vorne durchlaufen. $P_{i,j}$ wird vor die erste Aktion mit gleichem i und einem größeren j eingereiht. Existieren keine Aktionen mit gleichem i und größerem j, wird $P_{i,j}$ vor die erste Aktion mit einem größeren i eingereiht. Existiert auch keine Aktion mit einem größeren i, so wird $P_{i,j}$ am Ende der Warteschlange eingefügt. Die Aktionen der Warteschlange erfüllen zu jedem Zeitpunkt die Invariante:

$$P_{i,j} \text{ vor } P'_{i',j'} \quad \text{gdw.} \quad i < i' \ \vee \ (\ i = i' \ \wedge \ j < j' \ )$$

Für die *Tiefensuche* gilt: Die Warteschlange wird von vorne durchlaufen. $P_{i,j}$ wird vor die erste Aktion mit einem größeren j eingereiht. Beim beschriebenen Strukturmodell (vgl. Abbildung 9.2) kann keine Aktion mit demselben j in der Warteschlange sein. Existieren keine Aktionen mit größerem j, wird $P_{i,j}$ am Ende der Warteschlange eingefügt. Die Aktionen der Warteschlange erfüllen zu jedem Zeitpunkt die Invariante:

$$P_{i,j} \text{ vor } P'_{i',j'} \quad \text{gdw.} \quad j < j'$$

Sowohl die Breitensuche als auch die Tiefensuche werden durch die angegebenen Verfahren nur dann strikt durchgeführt, wenn die Warteschlange von genau einem Kontrollfaden bearbeitet wird. Sobald mehrere Kontrollfäden eingesetzt werden, kann die effektive Ausführungsreihenfolge leicht von der angestrebten Reihenfolge abweichen. Dafür wird der gegebene Parallelitätsgrad möglichst gut ausgenutzt. Falls die gemäß der angestrebten Reihenfolge „nächste“ Aktion noch nicht ausgeführt werden kann, werden andere Aktionen sozusagen vorgezogen, um die zur Verfügung stehenden parallelen Verbindungen auszunutzen. Sobald die „nächste“ Aktion dann ausführbar wird, wird sie auch sofort vom nächsten frei werdenden Kontrollfaden bearbeitet.

Weiterhin sind auch *Mischformen* zwischen der Breiten- und Tiefensuche denkbar. Beispielsweise kann für die erste Informationsebene die Breitensuche und für alle weiteren Informationsebenen die Tiefensuche angewendet werden. Oder es kann solange die Strategie der Tiefensuche verfolgt werden, bis zu jedem Dokument hinreichend viele relevante Informationen vorhanden sind. Auf diese Weise kann der Wechsel zwischen Tiefen- und Breitensuche sogar dynamisch geschehen, d.h. in Abhängigkeit vom jeweiligen Dokument.

Um derartige Mischformen auf der Basis des vorgestellten Warteschlangenmodells realisieren zu können, d.h. um neue Aktionen in der Warteschlange an den entsprechenden Stellen positionieren zu können, ist die Berücksichtigung der Indizes der neuen Aktionen nicht mehr ausreichend. Zur Umsetzung können den einzelnen Aktionen (pro Stufe aber auch dynamisch und individuell) Werte zugewiesen werden, die die Priorität der Bearbeitung widerspiegeln.

Eine Aktion kann dann gemäß ihrer Priorität an die entsprechende Stelle der Warteschlange eingefügt werden. Die Warteschlange ist demnach immer nach den Prioritäten der Aktionen sortiert, so dass die Aktionen mit der höchsten Priorität zuerst ausgeführt werden.

9.6 Resümee

In diesem Kapitel wurde ein Lösungsansatz für die Anforderungen aus dem Bereich der Anfragebearbeitung erarbeitet, der aus drei Hauptkomponenten besteht. Die Komponente der Anfrageplanung erstellt zu einer gegebenen MLS-QL-Anfrage einen korrekten Anfrageplan mit einer Reihe von Steuerinformationen für die anderen beiden Komponenten. Die Komponente der Anfrageausführung führt die notwendigen Interaktionen mit den Wandlern der betreffenden Literaturdienste durch und gewährleistet dabei die Einhaltung der Lastvorgaben. Die Komponente der Ergebnisintegration führt die kontinuierlich gelieferten Teilergebnisse in einer integrierten Ergebnismenge zusammen und unterstützt dabei Filter-, Duplikaterkennungs-, Gruppier- und Sortieroperatoren.

Anschließend wurden die Handlungsspielräume bei der Realisierung der drei Komponenten identifiziert und mit unterschiedlichen Strategien besetzt. Ziel dabei war es, Strategien zu entwickeln, mit denen die Ergebnisse entweder schneller oder relevante Ergebnisse früher beschafft werden können. Die Komponente der Anfrageplanung kann eine gegebene Anfrage nahezu ohne Veränderungen an die Literaturdienste weiterleiten oder aber mehrere Teilanfragen generieren, die insgesamt dieselben Ergebnisse wie die gegebene Anfrage liefern. Die Strategie der präferenzbasierten Anfragezerlegung wurde in erster Linie dafür entwickelt, um relevante Ergebnisse früher beschaffen zu können. Für die Komponente der Anfrageausführung wurden verschiedene Strategien zur Bestimmung der Reihenfolge der Einzelaktionen eines Anfrageplanes entwickelt (FCFS, Breiten- und Tiefensuche sowie Mischformen). Die Strategien können sowohl mit dem Zeitverhalten der vorliegenden Dienste als auch mit den Vorlieben des Benutzers in Einklang gebracht werden, so dass die Ergebnisse schneller und auch relevante Ergebnisse früher beschafft werden können. Im folgenden Kapitel werden für die Komponente der Ergebnisintegration geeignete Strategien für den Bereich der Duplikaterkennung entwickelt.

Auch wenn in diesem und dem nächsten Kapitel bereits einige theoretische Überlegungen über die Vorteile und Einsatzmöglichkeiten der unterschiedlichen Strategien angestellt wurden bzw. werden, lässt sich die tatsächliche Wirksamkeit der Strategien nur im praktischen Einsatz bestimmen. Daher werden in Kapitel 11 Bewertungsmaße für das Antwortzeitverhalten eines Meta-Recherchesystems entwickelt, damit im dritten Teil der Arbeit Experimente mit den unterschiedlichen Strategien durchgeführt werden können. Durch die Erprobung der Strategien in unterschiedlichen Szenarien kann ihre Wirkung im Hinblick auf das Antwortzeitverhalten charakterisiert und umfassend beurteilt werden.

10 Strategien der Duplikaterkennung

10.1 Überblick

In diesem Kapitel soll der Handlungsspielraum bei der Duplikatbehandlung durch unterschiedliche Strategien zur Bestimmung der Duplikate besetzt werden. Dazu wird zunächst noch einmal das grundlegende Problem konkretisiert, das im Wesentlichen darin besteht, dass es keine einheitliche bzw. eindeutige Interpretation für den Begriff des Duplikats im Bereich der Literatur gibt (Abschnitt 10.2). Dadurch kann auch keine geradlinige algorithmische Umsetzung der Duplikaterkennung erfolgen. In Abschnitt 10.3 wird der vorgeschlagene Lösungsansatz, der bereits im Rahmen der MLS-QL-Spezifikation angedeutet wurde (Abschnitt 6.4), detaillierter ausgearbeitet, insbesondere im Hinblick auf die praktische Realisierung. Der Handlungsspielraum soll mit unterschiedlichen Interpretationen von Duplikaten und unter dem Einsatz von unterschiedlichen Algorithmen besetzt werden, wobei das Ziel stets darin besteht, möglichst viele vorhandene Duplikate zu identifizieren und den Ergebnismengenaufbau durch die Duplikaterkennung möglichst wenig zu verzögern. Für die Duplikaterkennung werden Algorithmen für tolerante Zeichenkettenvergleiche benötigt. In diesem Kapitel werden zwei der bekanntesten Algorithmen in geeigneter Weise für den vorgeschlagenen Lösungsansatz angepasst (Abschnitt 10.4 und 10.5). Darüber hinaus wird ein neuer Algorithmus entwickelt (Abschnitt 10.6), der die Vorteile der beiden bekannten Algorithmen vereint.

10.2 Problem

Im Rahmen eines Meta-Recherchesystems ist es das Ziel der Duplikaterkennung, die Dokumente, die bei unterschiedlichen Literaturdiensten bezogen werden können, zu identifizieren und zusammenzuführen, so dass der Benutzer bei einem Dokument zwischen unterschiedlichen Beschaffungskonditionen (Lieferzeit, Lieferpreis, Lieferadresse, ...) wählen kann.

Im Literaturbereich wird seit 1973 für jedes Buch eine Internationale Standardbuchnummer (ISBN) vergeben. Die ISBN, die jedes Buch eindeutig identifiziert, setzt sich zusammen aus der Länder- bzw. Sprachkennung (1-5 Ziffern, z.B. 3 für Deutschland, Österreich und die deutschsprachige Schweiz, 0 und 1 für alle englischsprachigen Länder, 87 für Dänemark), der Verlagsnummer (3-6 Ziffern), der Titelnummer (2-5 Ziffern) und einer Prüfziffer. Die einzelnen Zifferngruppen werden durch Bindestriche voneinander getrennt, z.B.

3-453-20913-3
3-446-21368-6
3-930673-72-X

Wenn man die Bindestriche weglässt, besteht eine ISBN stets aus genau 10 Ziffern. Die letzte Ziffer (Prüfziffer) wird aus den ersten 9 Ziffern x_1, x_2, ..., x_9 berechnet:

$$x_{10} = (\sum_{i=1}^{9} (i * x_i)) \text{ modulo } 11$$

Dadurch können Einzelfehler und teilweise auch Doppelfehler bei der Eingabe erkannt werden. Da die Prüfziffer mittels modulo 11 berechnet wird, reichen die Ziffern 0 bis 9 nicht aus. Für die Zahl 10 wird die römische Zehn 'X' verwendet. Jeder Verlag besitzt eine eindeutige Verlagsnummer (z.B. Springer Verlag in Deutschland 540). Die Verlagsnummer richtet sich nach dem Umfang der Verlagsproduktion, d.h. je höher die Titelproduktion, desto weniger Stellen hat die Verlagsnummer, damit die 10 Stellen der ISBN zur eindeutigen Beschreibung jedes Buches ausreichen.

Interpretation 1 des Duplikatbegriffs. Um bei einem Meta-Recherchesystem die Duplikate zu bestimmen, kann die ISBN-Angabe herangezogen werden. Bücher, die dieselbe ISBN aufweisen, werden als Duplikate identifiziert.

Diese Vorgehensweise hat zum einen den technischen Nachteil, dass die ISBN meist erst auf der zweiten Informationsebene der Literaturdienste geliefert wird (vgl. Tabelle 8.3). D.h. Duplikate können nur ermittelt werden, wenn diese Informationen vom Meta-Recherchesystem auch beschafft werden. Bestehende Meta-Recherchesysteme wie der KVK oder Daffodil können beispielsweise die Duplikaterkennung über die ISBN aufgrund ihres geringen Integrationsgrades prinzipiell nicht durchführen (vgl. Abschnitt 3.3.2). Die Duplikaterkennung mithilfe der ISBN-Angaben kann erst zu dem Zeitpunkt stattfinden, an dem die Informationen der zweiten Ebene eingetroffen sind. Das hat entweder zur Folge, dass die Ergebnisse erst wesentlich später angezeigt werden können als die Informationen der ersten Ebene eingetroffen sind oder dass sich die angezeigten Ergebnisse im Nachhinein noch ändern können (d.h. mit anderen Ergebnissen verschmolzen werden können).

Der zweite Nachteil der Duplikatbestimmung über die ISBN besteht darin, dass für neue Auflagen und verschiedene Ausgaben (z.B. gebundene Ausgaben oder Taschenbuchausgaben) desselben Buches unterschiedliche ISBN-Angaben vergeben werden, obwohl der Inhalt jeweils identisch oder nahezu identisch (bei erweiterten Auflagen) ist. Dabei kann man sich durchaus vorstellen, dass ein Benutzer verschiedene Auflagen oder Ausgaben eines Buches als gleichwertig betrachten möchte und unabhängig von den Buchvarianten in erster Linie an der besten Beschaffungsalternative interessiert ist. Eine derartige Interpretation von Duplikaten lässt sich nicht auf Basis der ISBN-Angaben realisieren.

Interpretation 2 des Duplikatbegriffs. Die Duplikate werden aufgrund von gleichen Titel- und gleichen Autoren-Angaben ermittelt. Dadurch werden Duplikate aufgrund desselben Inhalts (geringfügige Änderungen sind möglich) gebildet, wohingegen formale Aspekte eines Buches bis hin zum Publikationstyp (z.B. Buch oder Hörbuch) vernachlässigt werden können.

Die Duplikaterkennung aufgrund von gleichen Titel- und Autoren-Angaben weist ebenfalls einen entscheidenden Nachteil auf. Die bibliographischen Daten, die die Literaturdienste zur Beschreibung ihres Dokumentenbestandes führen, sind oft fehlerhaft oder weisen uneinheitliche Aufnahmeformate auf. D.h. ein exakter Zeichenkettenvergleich der Titel- und Autoren-Angaben ist zu restriktiv und kann nur eine Untermenge der vorhandenen Duplikate identifizieren. Dennoch arbeiten existierende Meta-Recherchesysteme bisher nur mit exakten Zeichenkettenvergleichen (vgl. Abschnitt 3.2.2).

Untersuchungen aus den 80er Jahren haben ergeben, dass in Katalogeinträgen mehr als 10 % Schreibfehler vorkommen, d.h. jedes zehnte Wort ist falsch geschrieben, wobei davon 80 % der Fehler durch Einfügen, Ändern, Ersetzen und Vertauschen einiger Zeichen zu beheben sind (vgl. http://www.ifi.unizh.ch/CL/gschneid/LexMorphVorl/Lexikon10.IR.html). Neben Schreibfehlern findet man bei den Katalogaufnahmen sowohl bei Titel- als auch bei Autoren-Angaben einen recht individuellen Umgang mit Satz- und Sonderzeichen, mit Abkürzungen und mit Auslassungen. Um mehr vorhandene Duplikate identifizieren zu können, sollte beim Titel- und Autoren-Vergleich der Zeichenketten eine gewisse Abweichung toleriert werden.

Implementierungen für tolerante Zeichenkettenvergleiche weisen generell eine höhere algorithmische Komplexität auf als exakte Zeichenkettenvergleiche. Eine Anforderung an die Duplikaterkennung ist aber (vgl. 9.2), dass sie die Lieferung der Ergebnisse möglichst wenig verzögert. Daher ist bei der Realisierung der Duplikaterkennung sowohl die Qualität – was die Erkennungsleistung der Duplikate betrifft – als auch die Schnelligkeit der Implementierung zu berücksichtigen und gegeneinander abzuwägen.

10.3 Lösungsansatz

Das Problem bei der Duplikaterkennung liegt in erster Linie darin, dass es keine einheitlich akzeptierte Interpretation des Duplikatbegriffs gibt. Neben den zwei extremen Interpretationen des Duplikatbegriffs, die im vorangegangenen Abschnitt genauer ausgeführt wurden, sind auch diverse Mischformen denkbar, die beispielsweise unterschiedliche Auflagen eines Buches als identisch, unterschiedliche Ausgaben oder gar Publikationstypen von Dokumenten allerdings nicht als identisch betrachten.

Zur Lösung der beschriebenen Duplikatproblematik wird ein flexibler Ansatz vorgeschlagen, bei dem beliebige Attribute des Domänenmodells zur Bestimmung der Duplikatsemantik festgelegt werden können. Dabei werden die Dokumente als Duplikate angesehen, deren Werte bei allen angegebenen Attributen übereinstimmen. Diese Vorgehensweise wurde bereits im Rahmen der MLS-QL-Spezifikation eingeführt (vgl. Abschnitt 6.4, bei der Spezifikation der DUPLICATES-Klausel). Die Übereinstimmung der Werte bei den angegebenen Attributen kann sowohl in exakter als auch in toleranter Weise gefordert werden.

Die Gestaltung des Handlungsspielraums bei der Duplikaterkennung betrifft also zwei Dimensionen: Zum einen können verschiedene Attributkombinationen gewählt werden und zum anderen können für die ausgewählten Attribute verschiedene Arten von Vergleichen verwendet werden: exakte und tolerante Vergleiche. Exakte Vergleiche können sowohl bei Text- als auch bei Zahlenangaben eingesetzt werden, tolerante Vergleiche sollen allerdings nur für Attribute angegeben werden, die längere Textangaben enthalten und bei denen sich leicht Schreibfehler oder Aufnahmevarianten einschleichen. Für die Realisierung von toleranten Zeichenkettenvergleichen sollen unterschiedliche Algorithmen bereitgestellt werden.

Die Evaluierung des vorgeschlagenen Lösungsansatzes und der realisierten Algorithmen soll im dritten Teil der Arbeit durch geeignete Experimente erfolgen. Dabei soll herausgefunden werden, welche Kombination von Attributen und Algorithmen sowohl möglichst gute Duplikaterkennungsraten als auch möglichst geringe Verzögerungen beim Ergebnismengenaufbau hervorbringen können. Für die Algorithmen der toleranten Zeichenkettenvergleiche soll dabei auch herausgefunden werden, welche Toleranzwerte für geeignete Duplikaterkennungsraten zu wählen sind.

Kein Handlungsspielraum besteht beim vorgeschlagenen Lösungsansatz dagegen beim Aufbau der Ergebnismenge. Dieser geschieht kontinuierlich und soll durch die Duplikaterkennung so wenig wie möglich verzögert werden. Die Duplikaterkennung soll nicht blockierend arbeiten, d.h. die Ergebnisse werden in der Reihenfolge, in der sie vom Meta-Recherchesystem in Empfang genommen werden, in die Ergebnismenge eingefügt. Jedes neu eintreffende Ergebnis wird mit allen bereits in der Ergebnismenge vorhandenen Ergebnissen verglichen und ggf. – wenn eine Übereinstimmung in allen geforderten Attributen besteht – mit einem bereits vorhandenen Ergebnis verschmolzen.

Der Aufwand für die Vergleichsoperationen beim Einfügen eines neuen Ergebnisses in die Ergebnismenge ist nicht zu vernachlässigen. Das erste eintreffende Ergebnis muss natürlich mit keinem vorhandenen Ergebnis verglichen werden. Das zweite eintreffende Ergebnis muss mit einem vorhandenen Ergebnis verglichen werden. Das setzt sich solange fort, bis schließlich das n-te (letzte) eintreffende Ergebnis mit n-1 vorhandenen Ergebnissen verglichen werden muss. Teilweise sind auch etwas weniger Vergleiche notwendig, wenn zuvor einige Ergebnisse im Zuge der Duplikatzusammenführung miteinander verschmolzen wurden. Im allgemeinen beträgt der Vergleichsaufwand beim Einfügen und Verschmelzen von n Ergebnissen in die Ergebnismenge

$$\sum_{i=1}^{n}(i-1)=\sum_{i=0}^{n-1}i=\frac{(n-1)}{2}*n$$

Bei der geschilderten Vorgehensweise ist der Aufwand für die durchzuführenden Vergleiche also quadratisch, d.h. O(n*n).

Werden zur Bestimmung der Duplikate exakte Vergleiche durchgeführt, kann der Aufwand, d.h. die Anzahl der durchzuführenden Vergleiche, reduziert werden. Man kann beispielsweise mit einer geeigneten Sortierung der bereits vorhandenen Ergebnisse arbeiten oder die vorhandenen Ergebnisse mit denselben Attributwerten gezielt über Hash-Strukturen aufgreifen.

Bei toleranten Zeichenkettenvergleichen können allerdings keine Hilfsstrukturen gebildet werden, die die Anzahl der durchzuführenden Vergleiche im Vorhinein beschränken können. Erst wenn der Ähnlichkeitswert zweier Zeichenketten explizit berechnet wurde, können Aussagen in Bezug auf die Übereinstimmung getroffen werden. Die Ähnlichkeitsrelation auf Zeichenketten ist nämlich nicht transitiv, d.h. wenn Zeichenkette *A* hinreichend ähnlich mit Zeichenkette *B* ist und Zeichenkette *B* hinreichend ähnlich mit Zeichenkette *C* ist, kann daraus nicht abgeleitet werden, dass Zeichenkette *A* hinreichend ähnlich mit *C* ist. Daher kann für die vorhandenen Ergebnisse in der Ergebnismenge auch keine Einteilung in Äquivalenzklassen gefunden werden, die die Anzahl der durchzuführenden Vergleiche reduzieren könnte.

Bei Duplikaterkennung wird daher folgendes Verfahren eingesetzt: Die gewünschte Semantik der Duplikate wird durch die angegebenen Attribute und die entsprechenden Vergleichsoperatoren festgelegt. Duplikate müssen in allen angegebenen Attributen exakt bzw. hinreichend übereinstimmen. Sobald bei zwei Ergebnissen auch nur in einem der Attribute eine Unterscheidung entdeckt wird, können die Ergebnisse keine Duplikate mehr sein, d.h. es kann auf weitere Vergleiche zwischen diesen zwei Ergebnissen verzichtet werden. Daher werden beim Vergleich von zwei Ergebnissen zuerst die Attribute getestet, für die ein exakter Gleichheitsoperator gewählt wurde. Nur wenn die Ergebnisse in diesen Attributen übereinstimmen, werden die Attribute betrachtet, für die tolerante Vergleichsoperatoren angegeben wurden. Mit dieser Heuristik können viele der aufwendigeren toleranten Vergleiche eingespart werden. Enthält die Duplikatsemantik mehrere Attribute mit exakten Vergleichsoperatoren, so wird das Attribut zuerst überprüft, welches die Ergebnisse am stärksten diskriminiert, d.h. für welches die meisten unterschiedlichen Werte (Äquivalenzklassen) existieren. Auch dadurch kann eine Nicht-Übereinstimmung früher und damit schneller aufgedeckt werden. Z.B. sind die Attribute Publikationstyp oder Sprache durch ihre überschaubaren Ausprägungen vermutlich schlechtere Diskriminatoren als die Jahreszahl. Genauere Aussagen über die Diskriminationseigenschaft von Attributen können allerdings auch erst in den anschließenden Experimenten gewonnen werden.

Die folgenden Abschnitte widmen sich nun verschiedenen Algorithmen für die Realisierung von toleranten Zeichenkettenvergleichen. Dazu werden zunächst zwei der bekanntesten Verfahren in diesem Bereich aufgegriffen [PPF95]: die Methode der Trigramm-Vergleiche (Abschnitt 10.4) und die Bestimmung des Editierabstands zweier Zeichenketten gemäß der Damerau-Levenshtein-Metrik (Abschnitt 10.5). Schließlich wird in Abschnitt 10.6 ein neuer Algorithmus entwickelt, der die Vorteile der beiden bekannten Algorithmen miteinander vereint.

10.4 Trigramm-Vergleiche

Unigramme, Bigramme, Trigramme oder allgemein n-Gramme sind Teilzeichenketten mit der Länge von einem, zwei, drei bzw. n Zeichen einer Zeichenkette. Die Ähnlichkeit zweier Zeichenketten wird aus der Anzahl der gemeinsamen n-Gramme und der Zahl der insgesamt vorhandenen n-Gramme berechnet. Üblicherweise werden dazu Bi- oder Trigramme verwendet (vgl. [Sal89], [PPF95]).

Um die Ähnlichkeit zweier Zeichenketten Z_1 und Z_2 zu berechnen, geht man folgendermaßen vor. Für beide Zeichenketten wird je eine Multimenge aller Trigramme angelegt. In einer Multimenge kann – im Gegensatz zu einer normalen Menge – ein Element mehrfach vorkommen. Sei M_1 die Multimenge der Trigramme von Z_1, M_2 die von Z_2. Dann enthält die Multimenge $D_1 = M_1 \setminus M_2$ alle Trigramme, die in Z_1 vorkommen, aber nicht in Z_2. Die Multimenge $D_2 = M_2 \setminus M_1$ enthält entsprechend alle Trigramme, die in Z_2 vorkommen, aber nicht in Z_1. Die Multimenge $V = M_1 \cup M_2$ enthält alle Trigramme, die in Z_1 und Z_2 vorkommen. Die Multimenge $D = D_1 \cup D_2$ enthält dann alle Trigramme, um die sich die beiden Zeichenketten Z_1 und Z_2 unterscheiden.

Beispiel. Sei Z_1 = „Bandsalat“ und Z_2 = „Blattrand“. Dann sehen die gerade beschriebenen Multimengen wie folgt aus:

M_1 = { BAN, AND, NDS, DSA, SAL, ALA, LAT }

M_2 = { BLA, LAT, ATT, TTR, TRA, RAN, AND }

D_1 = { BAN, NDS, DSA, SAL, ALA }

D_2 = { BLA, ATT, TTR, TRA, RAN }

V = { ALA, AND, AND, ATT, BAN, BLA, DSA, LAT, LAT, NDS, RAN, SAL, TRA, TTR }

D = { ALA, ATT, BAN, BLA, DSA, NDS, RAN, SAL, TRA, TTR }

Um nun die Ähnlichkeit zwischen den beiden Zeichenketten zu bestimmen, wird aus den beiden Multimengen V und D jeweils ein normierter Vektor v und d gebildet. Dazu werden die Vielfachheiten der Elemente der Multimenge in einem Vektor notiert. Die Reihenfolge der Vielfachheiten ist dabei unerheblich. Danach wird jeder Vektor gemäß der Euklidischen Norm normiert. Die Euklidische Norm berechnet sich folgendermaßen:

$$|v| = \sqrt{v_1^2 + \ldots + v_n^2} \quad \text{für } v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$$

Damit berechnet sich die Ähnlichkeit der beiden Zeichenketten wie folgt:

$$trigramm(Z_1 ; Z_2) = \frac{|d|}{|v|}$$

Dieses Maß gibt die Ähnlichkeit der beiden Zeichenketten an. Die Werte, die dabei entstehen können, liegen zwischen 0 und 1. Je größer die Übereinstimmung der beiden Zeichenketten ist, desto näher liegt der Wert an 0. Um zu bestimmen, ob die Ähnlichkeit der beiden Zeichenketten hinreichend groß ist, so dass die Zeichenketten sozusagen als identisch gelten können, benötigt man noch einen Schwellwert *S*. Damit betrachtet man zwei Zeichenketten Z_1 und Z_2, als gleich, wenn *trigramm*(Z_1 ; Z_2) kleiner als dieser Schwellwert ist. Sie gelten als unterschiedlich, wenn *trigramm*(Z_1 ; Z_2) größer als dieser Schwellwert ist.

Im vorangegangenen Beispiel ist *trigramm*(Z_1 ; Z_2) ≈ 0,745, das bedeutet, es besteht ein großer Unterschied zwischen den Zeichenketten. Geeignete Schwellwerte für tolerante Zeichenkettenvergleiche liegen etwa im Bereich 0,4 bis 0,1, wobei ein höherer Schwellwert jeweils eine größere Toleranz bedeutet.

Vor- und Nachteile. Der Trigramm-Vergleich kann Umstellungen von ganzen Wörtern und Satzteilen gut erkennen, da sich beim Vergleich nur an den Randstellen der Umstellungen unterschiedliche Trigramme ergeben. Das kann beispielsweise bei Autorenangaben sehr nützlich sein, wenn die Autorennamen in unterschiedlicher Reihenfolge aufgeführt werden. Weiterhin sind Trigramm-Vergleiche gut geeignet für den Umgang mit Abkürzungen und Teilzeichenketten. Einzelne Tippfehler von nur einem Zeichen wirken sich allerdings immer gleich auf drei Trigramme aus, daher sind Trigramm-Vergleiche dafür eher nicht geeignet.

Aufwand. Gegeben sind zwei Zeichenketten Z_1 und Z_2 mit Länge n und m. O.B.d.A. sei n > m. Das Erzeugen der benötigten Trigramm-Mengen M_1 und M_2 hat einen linearen Aufwand, die Mengen M_1 und M_2 werden aus n+2 bzw. m+2 Trigrammen gebildet. Die anschließenden Mengenoperationen können ebenfalls mit linearem Aufwand durchgeführt werden, sofern die Trigramme in den Mengen in einer aufsteigenden Sortierung vorliegen. Damit entspricht der Aufwand der Trigramm-Vergleiche im Wesentlichen dem Sortieraufwand, also O(n log n).

10.5 Vergleiche mit der Damerau-Levenshtein-Metrik

Vergleiche von zwei Zeichenketten mit der Damerau-Levenshtein-Metrik basieren auf der Bestimmung des Editierabstands. Zur Bestimmung des Editierabstands muss zunächst festgelegt werden, welche Operationen dazu verwendet werden sollen. Im Falle der Damerau-Levenshtein-Metrik sind das die Operationen Einfügen, Löschen und Ersetzen. Die Operationen werden jeweils auf einzelne Zeichen angewandt. Zur Bestimmung der Ähnlichkeit zweier Zeichenketten wird die minimale Anzahl an benötigten Operationen berechnet, die zur Überführung der einen Zeichenkette in die andere benötigt wird.

Gegeben seien die beiden Zeichenketten Z_1 und Z_2, sei $z_1[\, i \,]$ das *i*-te Zeichen von Z_1, $z_2[\, j \,]$ das *j*-te Zeichen von Z_2, *m* die Länge von Z_1 und *n* die Länge von Z_2. Es soll nun die Zeichenkette Z_1 durch die genannten Operationen in die Zeichenkette Z_2 überführt werden.

Dazu wird nach folgenden Regeln rekursiv eine (m+1 × n+1)-Matrix M[0...m ; 0...n] aufgebaut:

$M[\ 0\ ;\ 0\ ] = 0$, $M[\ i\ ;\ 0\ ] = i$ für $1 \le i \le m$, $M\ [\ 0\ ;\ j\ ] = j$ für $1 \le j \le n$

$d(\ i\ ;\ j\ ) = 1$ falls $z_1[\ i\ ] \neq z_2[\ j\ ]$, $d(\ i\ ;\ j\ ) = 0$ falls $z_1[\ i\ ] = z_2[\ j\ ]$

$M[\ i\ ;\ j\ ] = \min \quad \{ \quad M[\ i\text{-}1\ ;\ j\ ] + 1, \quad$ = Löschen von $z_1[\ i\text{-}1\ ]$ in Z_1

$M[\ i\ ;\ j\text{-}1\ ] + 1, \quad$ = Einfügen von $z_2[\ j\text{-}1\ ]$ in Z_1

$M[\ i\text{-}1\ ;\ j\text{-}1\ ] + d(\ i\ ;\ j\ ), \quad$ = Ersetzen von $z_1[\ i\ ]$ durch $z_2[\ j\ ]$ in Z_1

$\}$

Für die beiden Zeichenketten “Bandsalat” und “Blattrand” ergibt sich damit die in Abbildung dargestellte Damerau-Levenshtein-Matrix.

		B	**A**	**N**	**D**	**S**	**A**	**L**	**A**	**T**
	0	1	2	3	4	5	6	7	8	9
B	1	0	1	2	3	4	5	6	7	8
L	2	1	1	2	3	4	5	5	6	7
A	3	2	1	2	3	4	4	5	5	6
T	4	3	2	2	3	4	5	5	6	5
T	5	4	3	3	3	4	5	6	6	6
R	6	5	4	4	4	4	5	6	7	7
A	7	6	5	5	5	5	4	5	6	7
N	8	7	6	5	6	6	5	5	6	7
D	9	8	7	6	5	6	6	6	6	7

Abbildung 10.1: Damerau-Levenshtein-Matrix

Die Anzahl der minimal benötigten Operationen ist dann $M[\ m\ ;\ n\ ]$. Dieser Wert wird durch die Länge der kleineren der beiden Zeichenketten geteilt und ist ein Maß für den Unterschied der beiden Zeichenketten. Der entstehende Wert liegt in der Regel zwischen 0 und 1, für das angegebene Beispiel bedeutet das $Dam\text{-}Lev(\ Z_1\ ;\ Z_2\ ) \approx 0{,}777$. Je näher der Wert bei 0 liegt, desto ähnlicher sind die beiden Zeichenketten. Der errechnete Wert wird – wie im vorangegangenen Abschnitt auch – mit einem geeignet zu wählenden Schwellwert verglichen.

Zeichenketten, deren Vergleichswert unter diesem Schwellwert liegen, werden als gleich betrachtet, andernfalls gelten die Zeichenketten als unterschiedlich.

Vor- und Nachteile. Der Damerau-Vergleich kann sehr gut mit Schreibfehlern umgehen, da diese jeweils nur eine zusätzliche Operation darstellen. Dabei ist es egal, ob ein Buchstabe ausgelassen, versehentlich eingefügt oder falsch eingegeben wurde. Da er aber – im Gegensatz zum Trigramm-Vergleich – die Zeichenketten positionsorientiert vergleicht, erkennt er Umstellungen von Worten oder Satzteilen nicht, da diese nicht durch die Operationen Einfügen, Löschen und Ersetzen abgedeckt werden können.

Aufwand. Wird der Algorithmus wie zuvor beschrieben implementiert, so hat er eine Komplexität von O(m*n). O.B.d.A. sei n > m. Der Aufwand kann allerdings noch deutlich reduziert werden, wenn im Vorhinein die Anzahl der erlaubten Fehler bekannt ist. Sei k die Anzahl der erlaubten Fehler, dann hat der Algorithmus eine Komplexität von O(k*n). k lässt sich aus dem gegebenen Schwellwert und der Länge der Zeichenketten ableiten. Der Algorithmus bricht sofort ab, wenn absehbar ist, dass die Fehlerrate zu hoch wird. Gerade im vorliegenden Szenario, wo die Zeichenketten sehr ähnlich sein sollen, die erlaubten Fehler also nur gering sein dürfen, ist eine derartige Implementierung von Vorteil, da k i.d.R. deutlich kleiner als m gewählt werden kann.

10.6 Block-Vergleiche

Beide bisher vorgestellten Verfahren haben klar erkennbare Vor- und Nachteile. Keiner der Vergleiche ist in der Lage, sowohl Schreibfehler als auch Umstellungen von Teilen der Zeichenketten zu erkennen und zu tolerieren. Für das vorliegende Literaturszenario wäre allerdings ein Vergleichsalgorithmus wünschenswert, der beide Fähigkeiten aufweist. Dazu wird in diesem Abschnitt ein neuer Vergleich vorgeschlagen und ausgearbeitet. Er ist an den Damerau-Levenshtein-Vergleich angelehnt, erweitert diesen aber sozusagen um die Operation „Umstellen einer Teilzeichenkette“.

Gegeben seien die beiden Zeichenketten Z_1 und Z_2, $z_1[\ i\ ]$ sei das i-te Zeichen von Z_1, $z_2[\ j\ ]$ das j-te Zeichen von Z_2, m die Länge von Z_1 und n die Länge von Z_2. Wie beim Damerau-Levenshtein-Verfahren wird auch hier zunächst eine Matrix aufgestellt, eine $m \times n$-Matrix M, die die einzelnen Werte allerdings nicht in addierter Form enthält, d.h.:

$$M[\ i\ ;\ j\ ] = 0 \text{ falls } z_1[\ i\ ] = z_2[\ j\ ]$$

$$M[\ i\ ;\ j\ ] = 1 \text{ falls } z_1[\ i\ ] \neq z_2[\ j\ ]$$

Abbildung 10.2 zeigt die entstehende Matrix über den bekannten Zeichenketten „BANDSALAT“ und „BLATTRAND“. Diese Matrix wird im Folgenden auch Antivalenzmatrix genannt.

	B	A	N	D	S	A	L	A	T
B	0	1	1	1	1	1	1	1	1
L	1	1	1	1	1	1	0	1	1
A	1	0	1	1	1	0	1	0	1
T	1	1	1	1	1	1	1	1	0
T	1	1	1	1	1	1	1	1	0
R	1	1	1	1	1	1	1	1	1
A	1	0	1	1	1	0	1	0	1
N	1	1	0	1	1	1	1	1	1
D	1	1	1	0	1	1	1	1	1

Abbildung 10.2: Antivalenz-Matrix

Das Damerau-Levenshtein-Verfahren basiert prinzipiell auf derselben Matrix. Der Algorithmus sucht sich in dieser Matrix sozusagen einen Weg, der von dem Wert M[1 ; 1] links oben zum Wert M[m ; n] rechts unten führt. Die Werte der Matrix, die dabei überschritten werden, werden addiert. Der Weg mit der kleinsten Summe ist die Vorschrift, um Z_1 in Z_2 mit der minimalen Anzahl von Operationen umzuwandeln, die Summe des Weges (geteilt durch die Länge der kürzeren Zeichenkette) ist das Maß für den Unterschied von Z_1 und Z_2.

Betrachtet man die Antivalenz-Matrix genauer, so fallen quadratische Teilmatrizen auf, deren Diagonalen nur 0-Werte enthalten (in Abbildung 10.2 grau markiert). Diese Teilmatrizen markieren Teilzeichenketten, die in beiden Zeichenketten vorkommen. Die Idee des vorgeschlagenen Algorithmus besteht nun darin, bei der Suche nach einer Überführung von Z_1 in Z_2 diese Teilmatrizen (Blöcke) in besonderem Maße zu berücksichtigen. Der Algorithmus wird im Folgenden als Block-Algorithmus und das Verfahren allgemein als Block-Vergleiche bezeichnet.

Der Block-Algorithmus bestimmt zunächst alle Teilmatrizen und deren Teilzeichenketten, die sowohl in Z_1 als auch in Z_2 vorkommen. Eine Teilmatrix bzw. ein Block ist jeweils der größtmögliche quadratische Ausschnitt der Antivalenz-Matrix, der in der Diagonalen nur 0-Werte beinhaltet.

Im nächsten Schritt wird versucht, die kleinere der beiden Zeichenketten möglichst vollständig durch die Teilzeichenketten der identifizierten Blöcke abzudecken. Jede Überdeckung lässt sich bewerten, indem man die Kosten für sie bestimmt. Für jeden

verwendeten Block in der Überdeckung wird 1 zu den Kosten addiert, für jedes nicht abgedeckte Zeichen wird ebenfalls 1 addiert.

Um eine geeignete Überdeckung zu finden, kann nun jede mögliche Kombination der gefundenen Blöcke mir den entsprechenden Kosten ausgewertet werden. Prinzipiell können sich die Zeichenketten der gefundenen Blöcke auch überschneiden. Bei k gefundenen Blöcken würde die Anzahl der möglichen Überdeckungen 2^k betragen.

Um diesen Aufwand zu verkleinern, wird nicht die optimale Überdeckung berechnet, sondern folgende Heuristik angewandt: Die Blöcke werden so bearbeitet, dass es im Hinblick auf die kleinere der beiden Zeichenketten keine Blöcke mehr gibt, die sich überlappen. Dazu wird jeweils vom kleineren der beiden Blöcke der Teil abgeschnitten, den er sich mit dem größeren Block teilt. Blöcke, die eine Größe von zwei unterschreiten, werden nicht berücksichtigt, da sie die gleichen Kosten wie ein nicht abgedecktes Zeichen verursachen. Dadurch ist die Verwendung jedes Blocks billiger, als wenn er nicht verwendet wird. In der Konsequenz werden alle Blöcke in der Überdeckung verwendet und ein Vergleich von mehreren Möglichkeiten der Überdeckung entfällt.

Die Kosten der Überdeckung ergeben sich dann als die Summe der verwendeten Blöcke und der nicht abgedeckten Zeichen (im verwendeten Beispiel wären die Kosten also 5). Dieser Wert wird durch die Länge der kleineren der beiden Zeichenketten geteilt und ist ein Maß für den Unterschied der beiden Zeichenketten, d.h. $Block(Z_1 ; Z_2) \approx 0{,}555$. Dieser Wert wird – wie bei den vorangegangenen beiden Verfahren auch – mit einem Schwellwert verglichen. Die beiden Zeichenketten gelten als gleich, wenn der Schwellwert unterschritten wird, andernfalls werden sie als unterschiedlich bewertet.

Vor- und Nachteile. Der Block-Vergleich kann sowohl Tippfehler als auch Umstellungen ganzer Wörter bzw. Satzteile tolerieren. Darüber hinaus ist der Block-Vergleich besonders gut geeignet, wenn eine Zeichenkette eine Teilzeichenkette der anderen Zeichenkette ist, wie das z.B. bei Abkürzungen oder Unterschlagungen von Autorennamen oder Untertiteln vorkommen kann. Ist allerdings eine Zeichenkette Z_1 viel größer als die andere Zeichenkette Z_2, dann ist es anzunehmen, dass die Teilzeichenketten von Z_1 die gesamte Zeichenkette Z_2 überdecken und so nur wenig Kosten verursacht werden, da keine einzelnen Zeichen unabgedeckt bleiben. Dieser Effekt, dass die kleine Zeichenkette durch zufällige kleine Teilzeichenketten der größeren bereits völlig abgedeckt wird, kann vermindert werden, indem bei der anfänglichen Bestimmung der Blöcke nur solche mit einer gewissen Mindestgröße zugelassen werden (z.B. mindestens 3x3-Blöcke). Dadurch wird verhindert, dass sich eine kleine Zeichenkette aus der größeren durch sehr kleine Teilzeichenketten „zusammenstückeln“ lässt, die gar keine gleichen Worte oder Satzteile der beiden Zeichenketten sind.

Aufwand. Die Komplexität des Block-Algorithmus beträgt O(n*m). Zur Bestimmung der Blöcke wird ausgehend von jedem Matrizeneintrag mit dem Wert 0 der größtmögliche Block identifiziert. Mit der vorgeschlagenen Heuristik kann die Überdeckung mit den identifizierten Blöcken in linearer Zeit berechnet werden.

10.7 Resümee

In diesem Kapitel wurde zunächst die Problematik der unterschiedlichen Interpretationen von Duplikaten umrissen. Daraufhin wurde der vorgeschlagene Lösungsansatz vorgestellt, dessen zentrale Idee darin besteht, zur Beschreibung der gewünschten Duplikatsemantik flexible Kombinationen von Attributen des Domänenmodells zuzulassen. Für jedes Attribut kann ein exakter oder ein toleranter Vergleichsoperator gefordert werden. Um den Aufwand der Duplikaterkennung beim Aufbau der Ergebnismenge zu reduzieren, wurde eine Heuristik vorgeschlagen, die bei den Ergebnisvergleichen zunächst Attribute mit exakten Vergleichen auswertet und dann erst die aufwendigeren toleranteren Vergleiche durchführt.

Anschließend wurde der Handlungsspielraum bei der Realisierung der toleranten Zeichenkettenvergleiche mit drei unterschiedlichen Verfahren besetzt. Dazu wurden zwei der bekanntesten Verfahren herangezogen (Trigramm- und Damerau-Levenshtein-Vergleiche) und ein neues Verfahren (Block-Vergleiche) entwickelt. Die Verfahren unterscheiden sich sowohl in ihrem Erkennungs- bzw. Toleranzverhalten als auch im Aufwand einer entsprechenden Implementierung. Trigramm-Vergleiche haben ihre Stärke im Berücksichtigen von Umstellungen, Damerau-Levenshtein-Vergleiche tolerieren gut Tippfehler von einzelnen Zeichen. Die Methode der Block-Vergleiche wurde entwickelt, um sowohl Tippfehler als auch Umstellungen und Teilzeichenketten berücksichtigen zu können. Details zur Implementierung der entsprechenden Algorithmen findet man in [Obe02].

Um einen ersten Eindruck vom Aufwand der Algorithmen zu bekommen, wurden in diesem Kapitel einige theoretische Überlegungen zur Komplexität der Verfahren angestellt. Im dritten Teil der vorliegenden Arbeit sollen Experimente konkrete Aussagen darüber geben, wie unterschiedlich die Qualität der Erkennungsleistung und die Beeinflussung des Antwortzeitenverhaltens der verschiedenen Algorithmen ist, vor allem auch im Zusammenspiel mit den anderen Komponenten der Anfragebearbeitung.

Das folgende Kapitel schließt mit der Entwicklung von geeigneten Bewertungsmaßen den zweiten Teil der Arbeit mit dem Schwerpunkt Anfragebearbeitung ab. Die Bewertungsmaße werden benötigt, um die in diesem und im vorangegangen Kapitel entwickelten Handlungsalternativen für die Komponenten der Anfragebearbeitung miteinander vergleichen und bewerten zu können.

11 Bewertungsmaße für die Anfragebearbeitung

11.1 Überblick

In den vorangegangenen Kapiteln wurden verschiedene Strategien für die Anfragebearbeitung und die Duplikaterkennung entwickelt. Um den Nutzen der verschiedenen Strategien beurteilen und bewerten zu können, werden in diesem Kapitel nun geeignete Bewertungsmaße entwickelt. Dazu werden zunächst noch einmal die Anforderungen zusammengestellt, die die Anfragebearbeitung eines Meta-Recherchesystems erfüllen soll (Abschnitt 11.2). Die Anfragebearbeitung soll möglichst alle Ergebnisse mit den zugehörigen Informationen in möglichst kurzer Zeit beschaffen. Die Idee des vorliegenden Ansatzes besteht nun darin, diese sich widerstrebenden Anforderungen aus der Sicht des Benutzers in Einklang zu bringen und zu bewerten. Schließlich entscheidet letztendlich auch der Benutzer darüber, ob ein Meta-Recherchesystem die gewünschte Funktionalität und eine akzeptable Performanz bietet.

In Abschnitt 11.3 werden die Bereiche des Meta-Recherchesystems identifiziert, die sich als Anknüpfungspunkte für die Bewertungsmaße eignen. Weiterhin werden Messgrößen erarbeitet, die für den Benutzer von Interesse und Bedeutung sind und daher Eingang in die Bewertungsmaße finden sollen. In Abschnitt 11.4 werden dann geeignete Bewertungsmaße entwickelt. Dazu werden zunächst eine Reihe von einzelnen Aspekten der Bewertung konkretisiert und anhand von Beispielen illustriert. Abschließend wird aus den berücksichtigten Einzelaspekten ein integriertes Bewertungsmaß abgeleitet, welches allgemein zur Bewertung des Antwortzeitverhaltens von Meta-Recherchesystemen vorgeschlagen wird.

11.2 Anforderungen und Lösungsansatz

Die Bewertungsmaße, die in diesem Kapitel entwickelt werden, sollen dazu dienen, die verschiedenen Strategien, die im Rahmen der Anfragebearbeitung eingesetzt werden können, beurteilen und miteinander vergleichen zu können. Für die Beurteilung und Bewertung sollen an dieser Stelle noch einmal die Anforderungen der Arbeit, die die Anfragebearbeitung betreffen, in Erinnerung gerufen werden (vgl. auch Abschnitt 9.2):

- A1: Einheitliche, integrierte Benutzungsschnittstelle
- A5: Vollständige Anfrageplanung und -bearbeitung
- A6: Duplikatbehandlung
- A7: Schnelle Antwortzeiten
- A8: Berücksichtigung von Lastvorgaben

Die wesentliche Herausforderung, die damit für die Anfragebearbeitung besteht, ist es, möglichst alle Ergebnisse (A5) mit den vorhandenen Zusatzinformationen (A1 und A5) in möglichst kurzer Zeit (A7) zu beschaffen. A6 und A8 können durch den vorgeschlagenen Lösungsansatz bereits als erfüllt betrachtet werden. Eine geeignete Duplikatbehandlung wurde in Kapitel 10 entwickelt. Sämtliche Strategien zur Reihenfolgebestimmung bei der Anfrageausführung (Abschnitt 9.5) berücksichtigen den vorgegebenen Parallelitätsgrad.

Zur Auflösung des Spannungsfeldes zwischen A1, A5 und A7 besteht die Idee des vorgeschlagenen Ansatzes darin, zusätzliche Anhaltspunkte aus der Sicht bzw. Wahrnehmung eines Benutzers heranzuziehen. Letztendlich machen nicht nur technische Kriterien (wie beispielsweise Ergebnisdurchsatz, d.h. gelieferte Ergebnisse pro Zeit) den Erfolg bzw. die Akzeptanz eines Meta-Recherchesystems aus, sondern auch die Zufriedenheit des Benutzers. Daher wollen wir hier noch einmal die Ergebnisse einiger Benutzerstudien zusammentragen, die anschließend bei der Spezifikation der Bewertungsmaße berücksichtigt werden sollen.

Im Rahmen des Anforderungskatalogs wurde bereits die allgemeine Ungeduld des Benutzers deutlich. [Nie94] belegt diese Ungeduld mit Zahlen. Antwortzeiten von mehr als 10 Sekunden sind für Benutzer nicht akzeptabel. Dem wurde in der vorliegenden Arbeit bereits dadurch Rechnung getragen, dass die eintreffenden Ergebnisse kontinuierlich angezeigt werden, um keine langen initialen Wartezeiten entstehen zu lassen. Dennoch sollte darauf geachtet werden, dass die Aufmerksamkeit des Benutzers mit zunehmender Zeit nachlässt, d.h. relevante Ergebnisse sollten möglichst früh präsentiert werden.

Bei der Begutachtung der Ergebnismenge wenden Benutzer bestimmte Entscheidungsstrategien an [Wan97]. Am häufigsten wird die Dominanz- und die Kombinationsstrategie verfolgt. Benutzer ziehen also entweder nur ein oder nur wenige ausgewählte Attribute für ihre Entscheidungsfindung heran (vgl. dazu Abschnitt 7.3.1). D.h. die im Rahmen des Domänenmodells verfügbaren Attribute sind für die Benutzer von unterschiedlicher Bedeutung, wobei diese Bedeutung sowohl zwischen verschiedenen Benutzern als auch beim selben Benutzer für verschiedene Anfragen variieren kann. Welche Attribute dabei am häufigsten als Entscheidungskriterium verwendet werden, wurde bereits in Abschnitt 5.3.2 dargestellt. Im vorliegenden Ansatz wird den Entscheidungsstrategien bereits durch die Unterstützung von Präferenzen Rechnung getragen.

Im folgenden Abschnitt werden zunächst die Alternativen bei der Festlegung der Messpunkte innerhalb des Meta-Recherchesystems diskutiert. Danach werden die zur Verfügung stehenden und geeigneten Messgrößen ermittelt, bevor in Abschnitt 11.4 dann die Spezifikation der Bewertungsmaße erfolgt.

11.3 Messpunkte und -größen im Meta-Recherchesystem

Ziel der Bewertungsmaße ist es, das Antwortzeitverhalten der unterschiedlichen Strategien im Rahmen der Anfragebearbeitung beurteilen und vergleichen zu können. Dazu sollen nun geeignete Ansatzpunkte beim vorliegenden Meta-Recherchesystem identifiziert werden. Abbildung 11.1 gibt dazu noch einmal einen Überblick über die Komponenten des Meta-Recherchesystems.

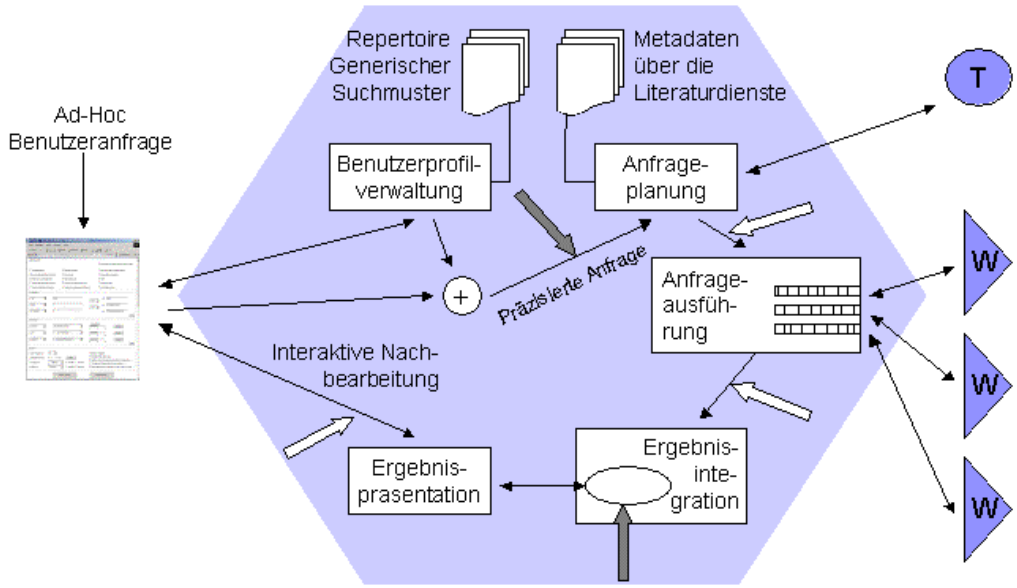


Abbildung 11.1: Mögliche (weiß) und gewählte (rot) Messpunkte im Meta-Recherchesystem

Die Wahl des Startpunkts für die Messungen des Antwortzeitverhaltens ist relativ naheliegend. Gewählt wird dazu der Zeitpunkt, an dem die Komponente der Anfrageplanung die gegebene MLS-QL-Anfrage erhält.

Durch die Messungen soll festgestellt werden, wie schnell dem Benutzer zu einer gegebenen Anfrage die Ergebnisse geliefert werden können. Dazu kann beispielsweise die Ergebnispräsentation überwacht werden und die Zeit ermittelt werden, wann die einzelnen Ergebnisse an der Benutzungsschnittstelle in Erscheinung treten. Allerdings hat die Komponente der Ergebnispräsentation lediglich die Aufgabe, die Ergebnisse, die in der integrierten Ergebnismenge vorliegen, in einer für den Benutzer geeigneten Weise aufzubereiten und darzustellen. D.h. die unterschiedlichen Strategien im Rahmen der Anfragebearbeitung haben einen direkten Einfluss auf den kontinuierlichen Ergebnismengenaufbau, aber die Zeit, die danach bis zur tatsächlichen Anzeige der Ergebnisse benötigt wird, hängt lediglich von der gewählten Präsentationsform und Realisierungstechnik der Benutzungsschnittstelle ab. Prinzipiell wäre es bei der vorliegenden Konzeption auch möglich, unterschiedliche Präsentationskomponenten für dieselbe Ergebnismenge anzubieten. Dadurch könnte

zusätzlich von den unterschiedlichen Präsentationskomponenten der Einfluss auf das Antwortzeitverhalten des Meta-Recherchesystems ermittelt werden. Davon wird im Rahmen der vorliegenden Arbeit allerdings abgesehen.

Damit ist zur Beurteilung der Leistung der Anfragebearbeitung beim vorliegenden Meta-Recherchesystem also der Zeitpunkt entscheidend, an dem die Ergebnisse in der integrierten Ergebnismenge vorliegen und somit (unabhängig von der gewählten Art der Ergebnispräsentation) für die weitere Verarbeitung zur Verfügung stehen. Für die Messungen wird folglich die kontinuierlich wachsende Ergebnismenge beobachtet.

Weiterhin besteht die Möglichkeit, zusätzliche Messpunkte beispielsweise zwischen den einzelnen Komponenten der Anfragebearbeitung einzufügen. Damit können unterschiedliche Strategien innerhalb derselben Komponente miteinander verglichen werden. Beispielsweise kann dadurch festgestellt werden, wie viel Zeit eine Anfrageplanung mit und ohne präferenzbasierter Anfragezerlegung in Anspruch nimmt oder wie unterschiedlich sich die Tiefen- und Breitensuche in der Komponente der Anfrageausführung verhalten. Allerdings kann vermutet werden, dass dabei keine bedeutsamen Unterschiede und Erkenntnisse gefunden werden. Signifikantere Unterschiede sind hingegen bei unterschiedlichen Strategien zur Duplikaterkennung zu erwarten. Diese können allerdings auch ohne zusätzliche Messpunkte ermittelt werden, indem beispielsweise dieselbe Anfrage einmal mit und einmal ohne Duplikaterkennung getestet und in ihren Auswirkungen auf den kontinuierlichen Ergebnismengenaufbau berücksichtigt wird.

Da es das Ziel der anschließenden Experimente sein wird, das Zusammenspiel der Anfragebearbeitungsstrategien zu beurteilen, werden keine zusätzlichen Messpunkte festgelegt. Wir betrachten lediglich die Auswirkungen auf den Ergebnismengenaufbau. Durch das Testen und Vergleichen von Anfragen mit unterschiedlichen Kombinationen von Anfragebearbeitungsstrategien ist es dennoch möglich, den Einfluss der einzelnen Strategien zu identifizieren.

Im Folgenden sollen nun Messgrößen erarbeitet werden, die zur Charakterisierung des kontinuierlichen Ergebnismengenaufbaus verwendet werden können. Zunächst kann beobachtet werden, wie die Dokumente (genauer gesagt die Dokumentreferenzen) in der Ergebnismenge in Erscheinung treten. Das kann durch eine Funktion beschrieben werden.

Anzahl Dokumente in der Ergebnismenge:

$$D : t \rightarrow D(t) \; ; \; D : IR_0^+ \rightarrow IN_0^+$$

Die Funktion $D(t)$ gibt die Anzahl der Dokumente in der Ergebnismenge zum Zeitpunkt t an. Die Funktion ist monoton steigend, da im Laufe der Zeit immer mehr Dokumente in der Ergebnismenge eintreffen. Der Wertebereich der Funktion ist ganzzahlig, da er die Anzahl der bereits vorhandenen Dokumente beschreibt. In dem Moment, in dem eines oder mehrere neue Dokumente in der Ergebnismenge eintreffen, macht die Funktion einen Sprung nach oben.

Daher ist $D(t)$ nicht stetig und auch nicht differenzierbar. Abbildung 11.2 gibt ein Beispiel für einen Kurvenverlauf.

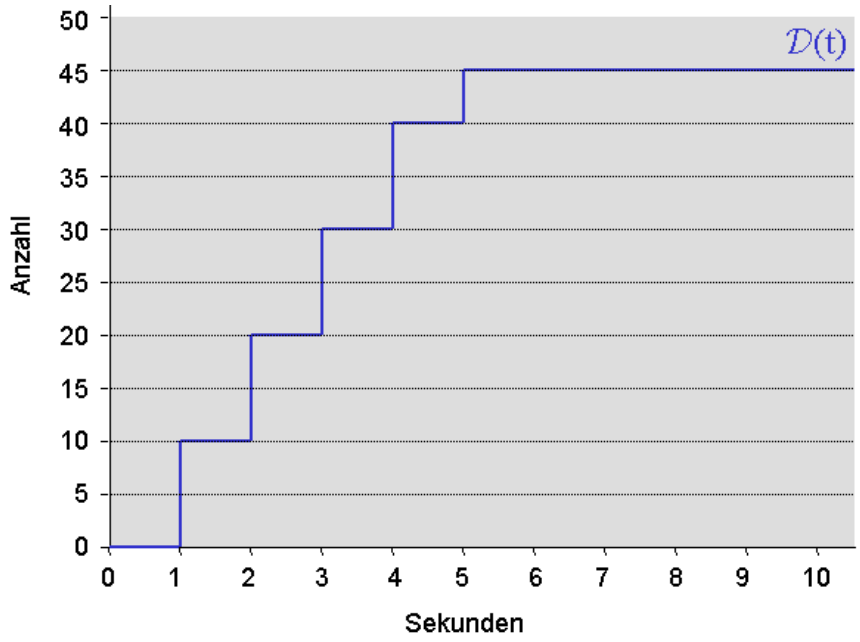


Abbildung 11.2: Beispiel für die Zunahme der Dokumente in der Ergebnismenge

Die in Abbildung 11.2 dargestellte Funktion $D(t)$ beschreibt ein fiktives Beispiel. Ein derartiger Kurvenverlauf ergibt sich, wenn ein Literaturdienst mit genau einer erlaubten parallelen Verbindung befragt wird, d.h. die Informationsportionen streng nacheinander bezogen werden. Dabei dauert die Beschaffung einer Informationsportion in diesem Beispiel jeweils genau 1 Sekunde. Insgesamt liefert die gegebene Anfrage 45 Dokumente. Am Verlauf der Kurve kann man sehen, dass die Ergebnisse immer in Zehnerportionen geliefert werden. Bei dem Literaturdienst liegt also eine horizontale Informationsportionierung vor, die jeweils zehn Kurztitel auf einer Ergebnisseite präsentiert. Ein Dokument gilt damit also als in der Ergebnismenge vorhanden, wenn die ersten Attribute bezogen worden sind. Das Eintreffen der weiteren Attribute spiegelt sich im Verlauf von $D(t)$ nicht wider.

Anzahl Attribute in der Ergebnismenge:

$$A : t \rightarrow A(t) \; ; \; A : IR_0^+ \rightarrow IN_0^+$$

Die Funktion $A(t)$ gibt die Anzahl der Attribute in der Ergebnismenge zum Zeitpunkt t an. Auch diese Funktion ist monoton steigend und aufgrund des ganzzahligen Wertebereichs nicht stetig und nicht differenzierbar. Im Kurvenverlauf der Funktion $A(t)$ spiegelt sich nun das Eintreffen von jeder Informationsportion wider, da in jeder Informationsportion eine Menge von Attributen vorhanden sind. Allerdings geht dabei der Bezug zu der Anzahl der eingetroffenen Dokumente verloren. Abbildung 11.3 zeigt zwei mögliche Kurvenverläufe.

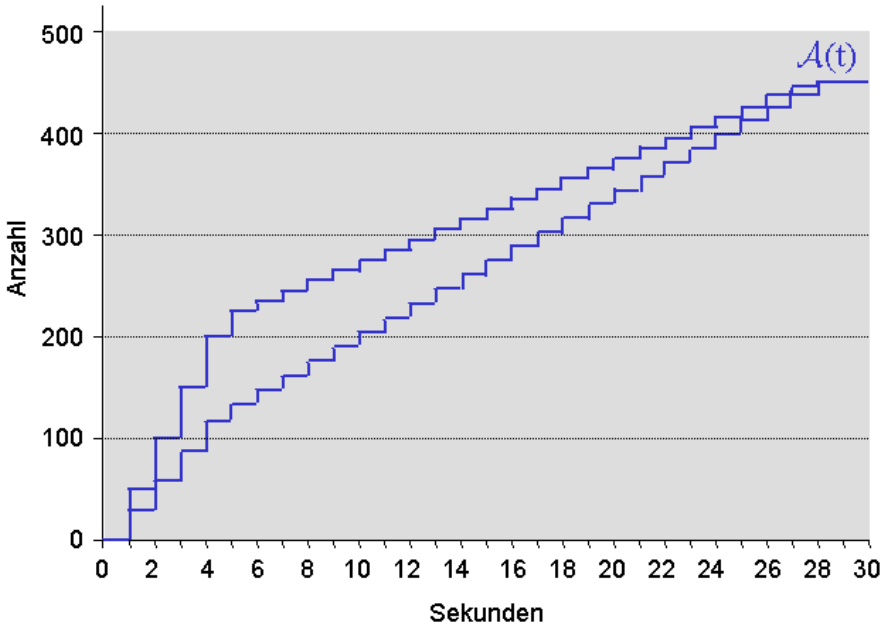


Abbildung 11.3: Beispiele für die Zunahme der Attribute in der Ergebnismenge

Die Kurvenverläufe in Abbildung 11.3 zeigen das portionsweise Eintreffen der Attribute, welches sich aufgrund der vertikalen Informationsportionierung der Literaturdienste ergibt. Die unterschiedlichen Kurven ergeben sich aufgrund einer unterschiedlichen vertikalen Informationsportionierung der befragten Dienste. Die obere Kurve gehört zu einem Buchhändler, die untere Kurve zu einem Bibliotheksdienst. Beide Dienste liefern – wie im vorangegangenen Beispiel auch – 45 Dokumente. Jedes Dokument wird insgesamt durch 10 Attribute beschrieben. Daher erreichen beide Kurvenverläufe auf dem Wert von 450 ihren Maximalwert. Beim Buchhandelskatalog werden auf der ersten und der zweiten Informationsebene jeweils 5 Attribute geliefert. Der Bibliotheksdienst liefert auf der ersten Informationsebene nur 3 Attribute, auf der zweiten Ebene dann 7 Attribute. Beide Dienste werden mit der Strategie der Breitensuche (vgl. Abschnitt 9.5) befragt. Es steht jeweils pro Dienst nur eine parallele Verbindung zur Verfügung, so dass die Ergebnisportionen nacheinander eintreffen. Als Antwortzeit für die Informationsportionen der ersten Ebene wird 1 Sekunde angenommen, die Portionen der zweiten Ebene können jeweils in einer halben Sekunde bezogen werden.

Die beiden unterschiedlichen Kurvenverläufe in Abbildung 11.3 machen deutlich, dass bei der Befragung des Buchhandelskatalogs vorübergehend mehr Informationen (d.h. Attribute) in der Ergebnismenge verfügbar sind als beim Bibliotheksdienst, obwohl beide Dienste insgesamt die gleiche Menge an Informationen liefern. Wäre als Ausführungsstrategie allerdings die Tiefensuche gewählt worden, wäre der Unterschied nicht so deutlich ausgefallen, da jeweils nach einer Informationsportion der ersten Ebene und zehn Informationsportionen der zweiten Ebene bei beiden Diensten genau dieselbe Anzahl von

Attributen verfügbar gewesen wäre. In Bezug auf die Dokumente würde sich für beide Dienste mit den genannten Vorgaben und der Strategie der Breitensuche jeweils die in Abbildung 11.2 dargestellte Kurve $D(t)$ ergeben.

Endzeitpunkt des Ergebnismengenaufbaus:

t_{max}

bezeichnet den Zeitpunkt, zu dem die gesamte Ergebnismenge vollständig aufgebaut ist, d.h. alle Dokumente sind mit allen verfügbaren bzw. allen angeforderten Attributen in der Ergebnismenge enthalten. Nach diesem Zeitpunkt ändert sich die Anzahl der Dokumente und der Attribute folglich nicht mehr.

$D = D(t_{max}) = D(t)$ für alle $t >= t_{max}$

$A = A(t_{max}) = A(t)$ für alle $t >= t_{max}$

In Abbildung 11.3 sind nach 29 Sekunden sämtliche Informationen vorhanden. Die Dokumente sind zwar schon früher eingetroffen (s. Abbildung 11.2, nämlich nach 5 Sekunden), aber die Beschaffung sämtlicher Attribute dauert länger, also $t_{max} = 28$, $D = 45$ und $A = 450$.

Da bekannt ist, dass Benutzer den verfügbaren Attributen eine unterschiedliche Bedeutung zumessen, sollten die Attribute nicht nur in ihrer Gesamtheit beobachtet werden. Beim Auftreten der Attribute soll zwischen bestimmten Attributen differenziert werden können. Daher benötigen wir noch folgende Messgrößen:

Vorkommen eines bestimmten Attributs in der Ergebnismenge:

$A_i : t \rightarrow A_i(t)$; $A_i : IR_0^+ \rightarrow IN_0^+$ wobei $i = 1, 2, \ldots, k$

$A_i = A_i(t_{max}) = A_i(t)$ für alle $t >= t_{max}$

Die Funktion $A_i(t)$ gibt an, wie oft das Attribut mit dem Index i in der Ergebnismenge zum Zeitpunkt t vorkommt. Dabei ergibt sich der Index i, indem jedem Attribut des Domänenmodells eindeutig eine Zahl zugeordnet wird. Bei dem vorliegenden Domänenmodell ist k=30 (vgl. Abschnitt 5.6).

Da Benutzer nicht nur an bestimmten Attributen (z.B. Sprache), sondern auch an bestimmten Ausprägungen, d.h. Werten („de“, „en“) der Attribute interessiert sind, definieren wir noch eine weitere Messgröße:

Vorkommen eines bestimmten Attributwerts in der Ergebnismenge:

$A_i^{Wert} : t \rightarrow A_i^{Wert}(t)$; $A_i^{Wert} : IR_0^+ \rightarrow IN_0^+$ wobei $i = 1, 2, \ldots, k$

$A_i^{Wert} = A_i^{Wert}(t_{max}) = A_i^{Wert}(t)$ für alle $t >= t_{max}$

Die Funktion $A_i^{Wert}(t)$ gibt an, wie oft das Attribut mit Index i und einem bestimmten Wert in der Ergebnismenge zum Zeitpunkt t vorkommt.

Mehrwertige Attribute des Domänenmodells erhalten wie einwertige Attribute genau einen Index. Betrifft die Bestimmung eines Attributwerts ein mehrwertiges Attribut, so werden sämtliche Ausprägungen des Attributs auf eine Übereinstimmung mit dem gegebenen Wert überprüft.

11.4 Spezifikation der Bewertungsmaße

Im vorangegangenen Abschnitt wurden die relevanten Messgrößen für das vorliegende Meta-Recherchesystem formal eingeführt und anhand eines fiktiven und einfachen Beispiels erläutert. In der Praxis, also bei realen Anfragen und Antwortzeiten, mehreren parallel befragten Diensten und auch mehreren zugelassenen parallelen Verbindungen pro Dienst kann davon ausgegangen werden, dass die Kurvenverläufe weniger charakteristisch und weniger leicht zu interpretieren sind. Im Grunde genommen werden sich allerdings stets monoton steigende Kurven ergeben, die zu einem gewissen Zeitpunkt dann ihren Maximalwert erreichen.

Die Unterschiede, die sich in den Kurvenverläufen widerspiegeln, rühren zum einen von den befragten Diensten selbst her, d.h. die Informationsportionierung und das Antwortzeitverhalten eines Dienstes schlägt sich im Kurvenverlauf nieder. Zum anderen beeinflussen unterschiedliche Strategien der Anfragebearbeitung die Kurvenverläufe. Das wurde bereits im vorangegangenen Abschnitt angedeutet, als festgestellt wurde, dass die Strategie der Tiefensuche zu anderen Kurvenverläufen führt als die der Breitensuche.

In diesem Abschnitt werden nun Bewertungsmaße entwickelt, mit denen verschiedene Kurvenverläufe beurteilt werden können. Damit soll es möglich sein, verschiedene Strategien bei der Anfragebearbeitung miteinander vergleichen und damit feststellen zu können, welche Art der Anfragebearbeitung die Ergebnisse so liefern kann, wie es vom Benutzer gewünscht bzw. bevorzugt wird. Da Benutzer in der Regel sowohl an schnellen Ergebnissen als auch an aussagekräftigen Informationen interessiert sind, können mit derartigen Maßen die widersprüchlichen Anforderungen (A1, A5, A7) in Einklang gebracht werden.

11.4.1 Der reine Attributdurchsatz

Ein naheliegender, eher technisch orientierter Ansatz besteht darin, den Attributdurchsatz zu bestimmen, d.h. wie viele Attribute durchschnittlich pro Zeiteinheit geliefert werden können. Dazu formulieren wir folgendes Maß M1:

M1: Attributdurchsatz = A/t_{max}

Dabei wird die Zeit berücksichtigt, die zum vollständigen Beschaffen der Ergebnisse benötigt wird (t_{max}) sowie die Menge der beschafften Attribute (A). In diesem Maß spiegelt sich allerdings nicht wider, ob zwischenzeitlich mehr Attribute verfügbar waren oder nicht, d.h. beide Kurven in Abbildung 11.3 haben denselben Attributdurchsatz, obwohl ein Benutzer

vermutlich zufriedener mit den Ergebnissen der oberen Kurve wäre, weil sie etliche Attribute früher liefern kann als die Bearbeitungsstrategie der unteren Kurve. Daher sollen sich die folgenden Maße mehr an der Wahrnehmung des Benutzers orientieren.

11.4.2 Ein einfaches Maß für Benutzerzufriedenheit

Es ist bekannt, dass Benutzer ungeduldig sind und die Ergebnisse gerne so früh wie möglich sehen wollen. Folgendes Maß trägt dieser Beobachtung Rechnung:

$$\text{M2:} \quad \sum_{t=0}^{t_{max}} A(t)$$

Mit M2 wird im Wesentlichen der Flächeninhalt der Kurve $A(t)$ zur x-Achse hin beschrieben. Man kann sich das Maß M2 anschaulich als Integralbildung vorstellen, mathematisch ist eine Integralbildung allerdings nicht möglich, da die Funktion $A(t)$ ja weder stetig noch differenzierbar bzw. integrierbar ist. Wendet man M2 auf beide Kurven in Abbildung 11.3 an, so würde die obere Kurve besser als die untere Kurve bewertet werden. Exakt ergeben sich bei der Berechnung von M2 Werte von 9120 und 7864.

11.4.3 Personalisierte Benutzerzufriedenheit (informationsorientiert)

Im vorangegangen Maß wurden alle Attribute gleich behandelt. Bekannt ist aber auch, dass Benutzer bestimmte Attribute für ihre Entscheidungsstrategien bevorzugen. Folgendes Bewertungsmaß erlaubt es, das Vorkommen eines bestimmten Attributs nach den persönlichen Vorlieben des Benutzers zu gewichten:

$$\text{M3:} \quad \sum_{t=0}^{t_{max}} \sum_{i=1}^{30} (w_i * A_i(t))$$

Alle 30 Attribute des Domänenmodells werden mit einem Gewicht w_i zwischen 0 und 1 versehen, welches die Bedeutung für den Benutzer ausdrückt. Die Gewichte können beispielsweise aus einer Benutzeranfrage in MLS-QL abgeleitet werden. Nur die Attribute, die der Benutzer auch im Ergebnis angefordert hat (SELECT), erhalten ein Gewicht, das größer als 0 ist. Die Attribute, für die Präferenzen, Sortierungen oder Gruppierungen angegeben wurden, erhalten im Verhältnis zu den anderen Attributen ein größeres Gewicht, da sie für den Benutzer offensichtlich von besonderer Bedeutung sind.

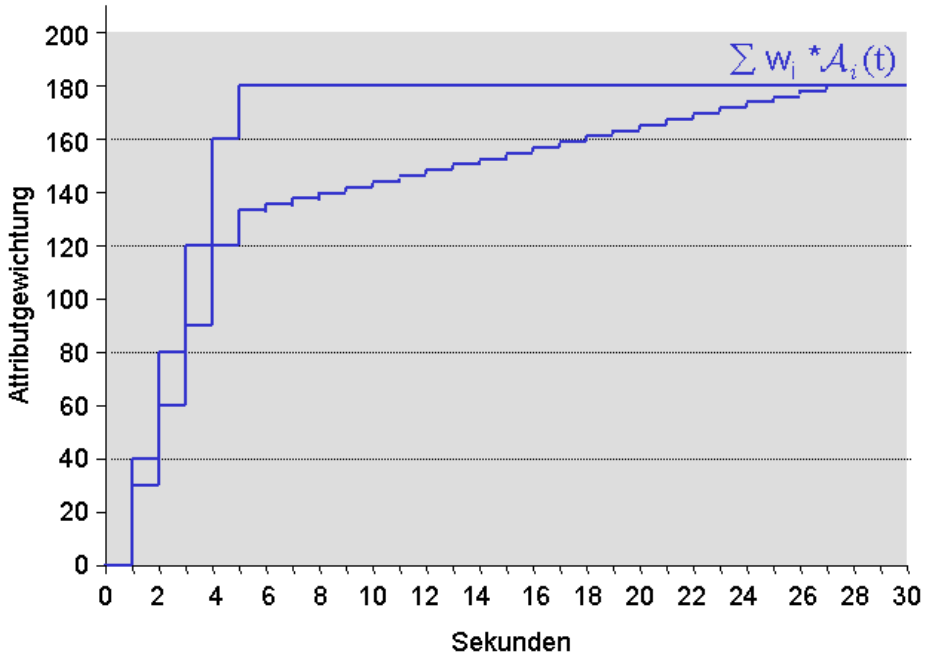


Abbildung 11.4: Kurvenverläufe mit gewichteten Attributen

Durch die Gewichtung der Attribute werden die ursprünglichen Kurvenverläufe $A(t)$ sozusagen in Bezug auf die y-Achse neu skaliert. Abbildung 11.4 zeigt dieselben Kurven wie in Abbildung 11.3, allerdings wurde eine persönliche Gewichtung verwendet. M3 berechnet dann wiederum den Flächeninhalt unter der gegebenen Kurve, nur dass die Kurve dafür in gewichteter Form vorliegt. In diesem Beispiel sind dem Benutzer Lieferzeiten sehr wichtig, genauso wichtig wie Titel, Autor und Jahr. Daher wurden diese vier Attribute mit dem Gewicht 1 versehen, die anderen Gewichte mit 0. Bei Buchhandelskatalogen (obere Kurve) wird die Lieferzeit gleich auf der ersten Ebene angegeben, daher erreicht die Kurve recht schnell ihren Maximalwert. Bei Bibliothekskatalogen befindet sich die Lieferzeit meist auf einer tieferen Ebene, in diesem Beispiel auf Ebene 2. Daher steigt die Kurve auch in der Phase, in der die Informationen der zweiten Ebene beschafft werden, weiterhin an.

Durch die Anwendung von M3 können also die Strategien der Anfragebearbeitung identifiziert werden, die die vom Benutzer bevorzugten Attribute möglichst früh liefern.

11.4.4 Personalisierte Benutzerzufriedenheit (wertorientiert)

Im vorigen Beispiel war der Benutzer an Informationen über die Lieferzeit, also dem Kriterium im Allgemeinen, interessiert. Der Benutzer kann aber auch insbesondere an Dokumenten mit bestimmten Werten interessiert sein. Ein Maß, welches lediglich konkrete Ausprägungen, also die bevorzugten Werte, begünstigt, kann folgendermaßen definiert werden:

$$\text{M4:} \quad \sum_{t=0}^{t_{max}} \sum_{i=1}^{30} \sum_{j=1}^{p_i} \left(w_{ij} * A_i^{Wert_j}(t) \right)$$

Für das Attribut mit Index i können auch mehrere Werte ($Wert_j$) mit unterschiedlichen Gewichten w_{ij} bevorzugt werden. Daher variiert die Obergrenze für die Anzahl der Werte (j) in Abhängigkeit vom jeweiligen Attribut (p_i). Dieses Maß hat eine starke Ähnlichkeit mit der Spezifikation von Präferenzen in MLS-QL (vgl. Abschnitt 6.6). Dadurch können die Strategien der Anfragebearbeitung identifiziert werden, durch die die präferierten Dokumente am schnellsten besorgt werden können.

11.4.5 Benutzerzufriedenheit von ungeduldigen Benutzern

Von Benutzern ist bekannt, dass sie nicht allzu lange auf das Eintreffen der Ergebnisse warten möchten. Daher kann entsprechend ein Maß definiert werden, welches die gelieferten Ergebnisse schon nach einer gewissen Zeit (x) bewertet und nicht wartet, bis alle Ergebnisse vollständig eingetroffen sind (t_{max}).

$$\text{M5:} \quad \sum_{t=0}^{x} A(t)$$

M5 bevorzugt diejenigen Strategien, die die meisten Attribute schon in der Anfangsphase beschaffen können. Dabei können für x beispielsweise 10 Sekunden, aber auch beliebige andere Werte gewählt werden. Wird x deutlich niedriger gewählt als t_{max}, so können plötzlich andere Strategien der Anfragebearbeitung besser bei der Bewertung abschneiden. Wenn die vollständige Beantwortung einer Anfrage beispielsweise länger als eine Minute dauert, so ist das Verhalten einer Anfragestrategie in der Zeit zwischen 50 und 70 Sekunden weit weniger bedeutend als in der Anfangsphase.

11.4.6 Ein integriertes Maß für Benutzerzufriedenheit

Die bisher vorgestellten Bewertungsmaße M2 - M5 zur Beschreibung der Benutzerzufriedenheit können nun zu einem integrierten Maß zusammengeführt werden. Dieses Bewertungsmaß berücksichtigt die in den letzten Abschnitten angesprochenen Aspekte:

$$\text{M6:} \quad \sum_{t=0}^{x} \sum_{i=1}^{30} \left(w_i * A_i(t) + \sum_{j=1}^{p_i} \left(w_{ij} * A_i^{Wert_j}(t) \right) \right)$$

M6 trägt damit den wesentlichen Erkenntnissen der Benutzerforschung Rechnung. Dabei wird die Funktion $D(t)$ zugunsten der aussagekräftigeren Funktionen, die die Attribute betreffen, vernachlässigt. Geeignete Belegungen für die Gewichte oder die Anfangszeit x werden im Rahmen der folgenden Experimente ermittelt werden (vgl. Kapitel 13).

11.5 Resümee

In diesem Kapitel wurden zunächst geeignete Messpunkte im vorliegenden Meta-Recherchesystem identifiziert. Weiterhin wurden Messgrößen ermittelt, die für den Benutzer von Interesse sind und für die Bewertung der Ergebnisse herangezogen werden können.

Anschließend wurden Bewertungsmaße entwickelt, die sich an der Wahrnehmung durch den Benutzer orientieren und sozusagen als Maß für eine Benutzerzufriedenheit interpretiert werden können. Durch diese Bewertungsmaße können die widersprüchlichen Anforderungen des Anforderungskatalogs, die die Vollständigkeit und die Schnelligkeit der gelieferten Ergebnisse betreffen, miteinander in Einklang gebracht werden. Anhand der Bewertungsmaße können nun verschiedene Strategien der Anfragebearbeitung beurteilt und miteinander verglichen werden. Eine absolute Beurteilung fällt auch mit den vorgeschlagenen Bewertungsmaßen insofern schwer, als sich jeweils nur ein relativ abstrakter Summen- bzw. Integralwert ermitteln lässt. Allerdings lassen sich die unterschiedlichen Strategien anhand der Bewertungsmaße gut miteinander vergleichen, so dass festgestellt werden kann, welche Strategie oder Kombination von Strategien eher den Bedürfnissen des Benutzers entspricht als andere.

Weiterhin spiegeln sich in den entwickelten Maßen auch die speziellen Eigenschaften der zugrundeliegenden Dienste wider, d.h. die Informationsportionierungen (vgl. Abschnitt 8.2), das Antwortzeitverhalten (vgl. Abschnitt 8.3) und der zugelassene Parallelitätsgrad. Dadurch können die in Kapitel 9 und 10 entwickelten Strategien zur Anfragebearbeitung zusätzlich dahingehend untersucht werden, bei welcher Art von Diensten und Anfragen sie den größten Nutzen bringen können und daher eingesetzt werden sollten.

Dieses Kapitel beschließt den zweiten Teil der Arbeit zum Schwerpunkt der Anfragebearbeitung. Der dritte und letzte Teil der Arbeit beschäftigt sich mit der Validierung des vorgeschlagenen Lösungsansatzes, d.h. mit der Entwicklung einer geeigneten Testumgebung (Kapitel 12) und der Durchführung der entsprechenden Experimente (Kapitel 13).

12 Die Simulationsumgebung SIMPSON

12.1 Überblick

Dieses Kapitel leitet den dritten und letzten Teil der vorliegenden Arbeit ein, der sich mit der Validierung des bisher vorgeschlagenen Lösungsansatzes beschäftigt. Im zweiten Teil der Arbeit wurden zunächst existierende Literaturdienste in Bezug auf ihre strukturellen Eigenschaften und ihr Antwortzeitverhalten untersucht (Kapitel 8). Daraufhin wurde in Kapitel 9 ein Lösungsansatz für die Konzeption und Umsetzung der Anfragebearbeitung eines Meta-Recherchesystems erarbeitet. Beim vorgeschlagenen Lösungsansatz konnten eine Reihe von Handlungsspielräumen identifiziert werden, die im weiteren Verlauf der Arbeit mit unterschiedlichen Strategien zur Anfrageplanung (Abschnitt 9.4), zur Anfrageausführung (Abschnitt 9.5) und zur Duplikaterkennung (Kapitel 10) besetzt wurden. Im vorangegangen Kapitel wurden schließlich Bewertungsmaße entwickelt, um unterschiedliche Verfahren bei der Anfragebearbeitung beurteilen und miteinander vergleichen zu können.

Ziel des dritten Teils der Arbeit ist es nun herauszufinden, welche Strategien der Anfragebearbeitung am besten geeignet sind bzw. in welchen Situationen und unter welchen Umständen welche Strategien der Anfragebearbeitung am besten geeignet sind. Dazu soll eine Reihe von Experimenten durchgeführt werden, die die einzelnen Strategien in verschiedenen Szenarien erprobt. Dabei besteht ein Szenario zum einen aus einer gegebenen Anfrage und zum anderen aus den Eigenschaften der zu befragenden Dienste, d.h. aus ihrer Strukturierung, ihrem Antwortzeitverhalten und den gemachten Lastvorgaben.

In diesem Kapitel soll eine Simulationsumgebung entwickelt werden, die die Durchführung der Experimente in geeigneter Weise unterstützt. Die zugrundeliegenden Dienste sollen in ihren strukturellen Eigenschaften und in ihrem Antwortzeitverhalten nachgebildet werden, um dadurch neben realen auch beliebige Szenarien berücksichtigen und testen zu können. Nach der Formulierung der Anforderungen an die Simulationsumgebung wird ein Lösungsansatz entwickelt und zunächst in seiner Grobarchitektur vorgestellt (Abschnitt 12.2). In den folgenden Abschnitten werden die Details des entwickelten Simulationswerkzeugs anhand eines durchgängigen Beispiels demonstriert (Abschnitte 12.3 - 12.6). Im noch folgenden Kapitel 13 erfolgt dann die Durchführung der Experimente zur Validierung der entwickelten Strategien zur Anfragebearbeitung in einem Meta-Recherchesystem.

12.2 Anforderungen und Lösungsansatz

12.2.1 Anforderungen

Ziel der Simulationsumgebung ist es, die Experimente zur Bewertung und Beurteilung der Anfragebearbeitungsstrategien in komfortabler Weise durchführen zu können. Die befragten Literaturdienste sollen dazu simuliert werden, damit auch Szenarien getestet werden können, die momentan nicht in der Realität zu finden sind. Beispielsweise sollen die Strategien der Anfragebearbeitung auch in Szenarien getestet werden, in denen die Dienste besonders schnell antworten oder in denen verhältnismäßig viel Last bzw. Parallelität erlaubt ist. Dadurch können Prognosen erstellt werden, inwiefern sich die Leistung der Anfragebearbeitung und damit auch eines Meta-Recherchesystems in Zukunft verändern wird, wenn die Antwortzeiten der Literaturdienste schneller werden und die Webserver der Literaturdienste mehr Last erlauben.

Im Folgenden sollen nun die Anforderungen an eine Simulationsumgebung formuliert werden. Das Simulationswerkzeug muss zunächst die Komponenten der Anfragebearbeitung mit sämtlichen zu testenden Strategien enthalten.

- Strategien der Anfrageplanung: Die Komponente der Anfrageplanung soll zu einer gegebenen MLS-QL-Anfrage sowohl einen Anfrageplan mit einer direkten Umsetzung der gegebenen Anfrage als auch einen Anfrageplan mit der Strategie der präferenzbasierten Anfragezerlegung erstellen können (vgl. Abschnitt 9.4).
- Strategien der Anfrageausführung: Die Komponente der Anfrageausführung soll einen gegebenen Anfrageplan mit den Strategien FCFS, Breitensuche und Tiefensuche abarbeiten können (vgl. Abschnitt 9.5).
- Strategien der Ergebnisintegration: Die Komponente der Ergebnisintegration soll beliebige Attributkombinationen zur Bestimmung der Duplikate berücksichtigen. Dabei sollen für die Attribute sowohl exakte als auch tolerante Vergleiche angegeben werden können. Als Strategien für tolerante Zeichenkettenvergleiche sollen Trigramm-Vergleiche, Damerau-Levenshtein-Vergleiche und Block-Vergleiche zur Verfügung stehen (vgl. Kapitel 10). Bei der Auswertung der Ergebnismenge soll deutlich werden, wie viele Dokumente von der jeweiligen Strategie als Duplikate erkannt wurden.

Weiterhin soll das Simulationswerkzeug die Zeit vom Abschicken einer gegebenen MLS-QL-Anfrage bis zum Eintreffen der Ergebnisse in der Ergebnismenge bestimmen. Dazu soll das kontinuierliche Eintreffen der Ergebnisse in der Ergebnismenge beobachtet werden.

- Bewertung: Beim kontinuierlichen Aufbau der Ergebnismenge soll die Zeit des Eintreffens von jedem Teilergebnis vermerkt werden. Das Wachstum der Ergebnismenge soll graphisch veranschaulicht werden. Zudem soll die Leistung der

Anfragebearbeitung anhand des integrierten Bewertungsmaßes M6 beurteilt werden (vgl. Kapitel 11). Die Attribute, die Werte, die Gewichte und die zu berücksichtigende Zeit sollen dabei explizit angegeben werden können.

Um die Literaturdienste in realistischer Weise simulieren zu können, müssen sowohl die strukturellen Eigenschaften als auch die Charakteristika des Antwortzeitverhaltens berücksichtigt werden.

- Dienstsimulation: Im Hinblick auf die Struktur der Literaturdienste müssen sowohl horizontale als auch vertikale Informationsportionierungen nachgebildet werden (vgl. Abschnitt 8.2). Weiterhin sollen die Literaturdienste ihre Ergebnisse auch in sortierter Form liefern können. Das Antwortzeitverhalten soll anhand des in Abschnitt 8.4 gefundenen Kurvenverlaufs (für die Verteilung der Antwortzeiten) nachgebildet werden. Das Antwortzeitverhalten soll für jede Informationsebene individuell konfiguriert werden können. Zur Konfiguration sollen die in Abschnitt 8.4 identifizierten fünf Parameter sowie eine graphische Veranschaulichung der Antwortzeitverteilung zur Verfügung stehen. Die simulierten Dienste sollen reale Ergebnisdaten zurückliefern, so dass die Duplikaterkennung auch auf einer realistischen Datengrundlage getestet werden kann. Die Ergebnisdaten können bereits in einem einheitlichen Format vorliegen, so dass im Rahmen der Simulationsumgebung auf den Einsatz von Wandlern verzichtet werden kann.

Die weiteren Anforderungen betreffen nun die Handhabung der Simulationsumgebung.

- Einfache Konfigurierbarkeit: Die Simulationsumgebung soll eine Benutzungsoberfläche bereitstellen, über die die möglichen Optionen und Parameterbelegungen auf einfache und übersichtliche Weise eingestellt werden können. Dies soll sowohl für die Wahl der Anfragestrategien gelten als auch für Bereitstellung der Anfrage, der Bewertungsfunktion und für die Konfiguration der Literaturdienste.
- Speicherung/Wiederverwendbarkeit: Die gewählten Einstellungen und Parameterbelegungen sollen für eine spätere Wieder- bzw. Weiterverwendung gespeichert werden können – genauso wie die Ergebnisse und die zugehörigen Auswertungen.
- Automatisierte Testdurchläufe: Die Simulationsumgebung soll nicht nur interaktiv bedient werden können, sondern auch die Möglichkeit bieten, eine Reihe von Testszenarien zusammenzustellen und diese dann automatisiert ablaufen zu lassen.

Nach der Zusammenstellung der Anforderungen wird im folgenden Abschnitt nun die Grobarchitektur der Simulationsumgebung vorgestellt.

12.2.2 Grobarchitektur

Um die Anforderungen des vorangegangenen Abschnitts erfüllen zu können, wird beim Entwurf der Simulationsumgebung folgende Vorgehensweise gewählt: Den Ausgangspunkt für die Simulationsumgebung bilden die Komponenten des Meta-Recherchesystems, die die Anfragebearbeitung durchführen. Das sind die Komponenten der Anfrageplanung, der Anfrageausführung und der Ergebnisintegration. Diese Komponenten enthalten jeweils die geforderten Bearbeitungsstrategien. Abbildung 12.1 zeigt im mittleren Bereich die bereits bekannten Komponenten der Anfragebearbeitung.

Die Komponenten der Anfragebearbeitung sollen nun mit einer MLS-QL-Anfrage und mit den gewünschten Anfragestrategien und Parametern angesteuert werden. Dazu wird eine neue Komponente mit der Bezeichnung Testsuite bzw. Test eingeführt. Ein Test beinhaltet sowohl die gegebene MLS-QL-Anfrage als auch die gewünschte Strategie der Anfrageplanung und Anfrageausführung. Die gewünschte Art der Duplikaterkennung ist bereits in der gegebenen MLS-QL-Anfrage enthalten. Eine Testsuite umfasst eine Menge von Tests, die in der angegebenen Reihenfolge automatisiert ausgeführt werden können. Die Daten eines Tests und einer Testsuite können in Form von Dateien abspeichert werden (Eingabeparameter). Diese Dateien werden als XML-Dokumente abgelegt, so dass sie sowohl mit als auch ohne die Simulationsumgebung in verständlicher Form lesbar sind.

In der gegebenen MLS-QL-Anfrage werden die zu befragenden Dienste festgelegt. Die Simulationsumgebung enthält eine weitere Komponente, die Komponente der Dienstsimulation, die sich mit der Nachbildung der strukturellen Eigenschaften und des Antwortzeitverhaltens der Literaturdienste beschäftigt. Für jeden Dienst existieren mehrere Mengen von Ergebnisdokumenten, die auf eine gegebene Anfrage hin präsentiert werden können – in der entsprechenden horizontalen und vertikalen Informationsportionierung und mit dem entsprechenden Antwortzeitverhalten. Über die Komponente der Dienstspezifikation können diese Eigenschaften für jeden Dienst festgelegt und abgespeichert werden (Diensteigenschaften). Jede Menge von Ergebnisdokumenten liegt als Datei vor. Eine Datei enthält jeweils eine Menge von XML-Dokumenten, die dem in Anhang C.1 spezifizierten Domänenmodell gehorchen. Die Dateien mit den entsprechenden Ergebnisdokumenten können erstellt werden, indem die realen Literaturdienste befragt, die Ergebnisse eingesammelt und in das vorgegebene Format gebracht werden.

Die Simulationsumgebung enthält eine weitere Komponente, die Komponente zur Ergebnisbewertung. Diese Komponente beobachtet das Eintreffen der Ergebnisse in der Ergebnismenge und notiert von jedem Teilergebnis den Zeitpunkt seines Eintreffens. Auf diesen Daten (die ebenfalls als Dateien abgespeichert werden können, Ergebnisse & Zeiten) basieren die zur Verfügung stehenden Bewertungen. Die Komponente der Ergebnisbewertung kann die kontinuierlich wachsende Ergebnismenge in textueller Form präsentieren oder in graphischer Form als monoton steigende Kurve visualisieren. Zusätzlich steht das integrierte

Bewertungsmaß zur Benutzerzufriedenheit (M6) zur Verfügung. Die benötigten Gewichte und Parameter dazu können über die Benutzungsschnittstelle eingegeben werden. Weiterhin wird bei der Ergebnisbewertung die Anzahl der erkannten Duplikate in der Ergebnismenge angegeben.

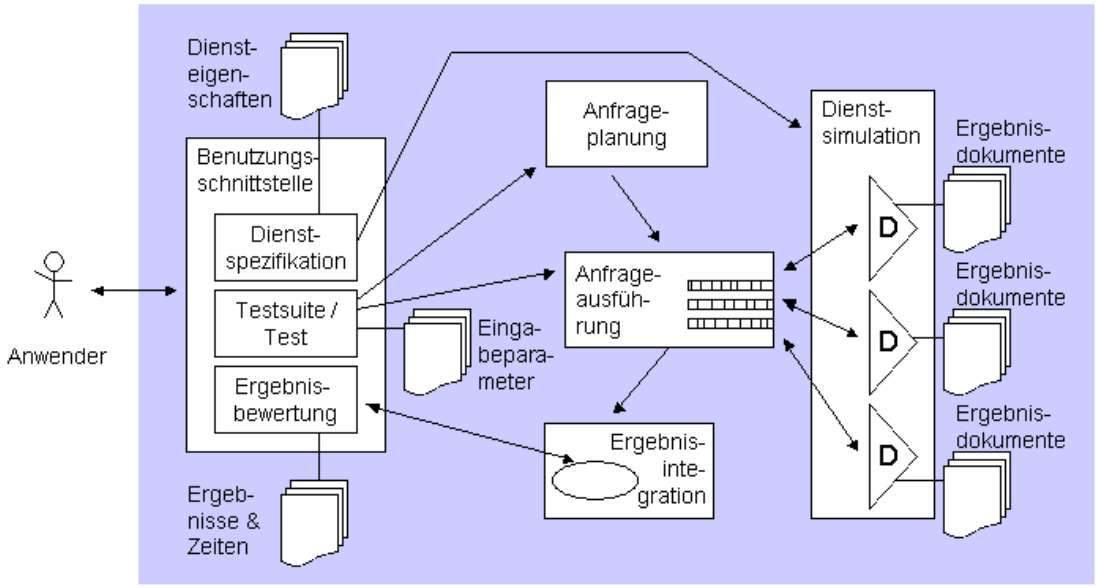


Abbildung 12.1: Grobarchitektur von SIMPSON

Die Komponenten der Simulationsumgebung, über welche der Anwender die Angaben, Strategien und Parameter der Anfragebearbeitung steuern und beeinflussen kann, werden in der umfassenderen Komponente der Benutzungsschnittstelle zusammengefasst. Abbildung 12.1 gibt einen Überblick über sämtliche Komponenten der Simulationsumgebung und deren Zusammenspiel. Das entwickelte Simulationswerkzeug wird im Folgenden SIMPSON genannt. Das Akronym SIMPSON steht dabei für a SIMulation tool for ProceSsing and OptimiziNg queries based on literature services on the Web [SO02b].

Abbildung 12.2 zeigt die Benutzungsschnittstelle von SIMPSON. Jede Komponente, über die Einfluss auf die Anfragebearbeitung genommen werden kann (Testsuite (links oben), Test (rechts oben), Dienstspezifikation (links unten), Ergebnisbewertung (rechts unten)), wird in einem separaten Fenster umgesetzt, in welchem alle notwendigen Angaben und Parameterbelegungen gemacht werden können. In den folgenden Abschnitten wird jede dieser Komponenten anhand eines durchgängigen Beispiels detaillierter vorgestellt (Abschnitte 12.3 - 12.6).

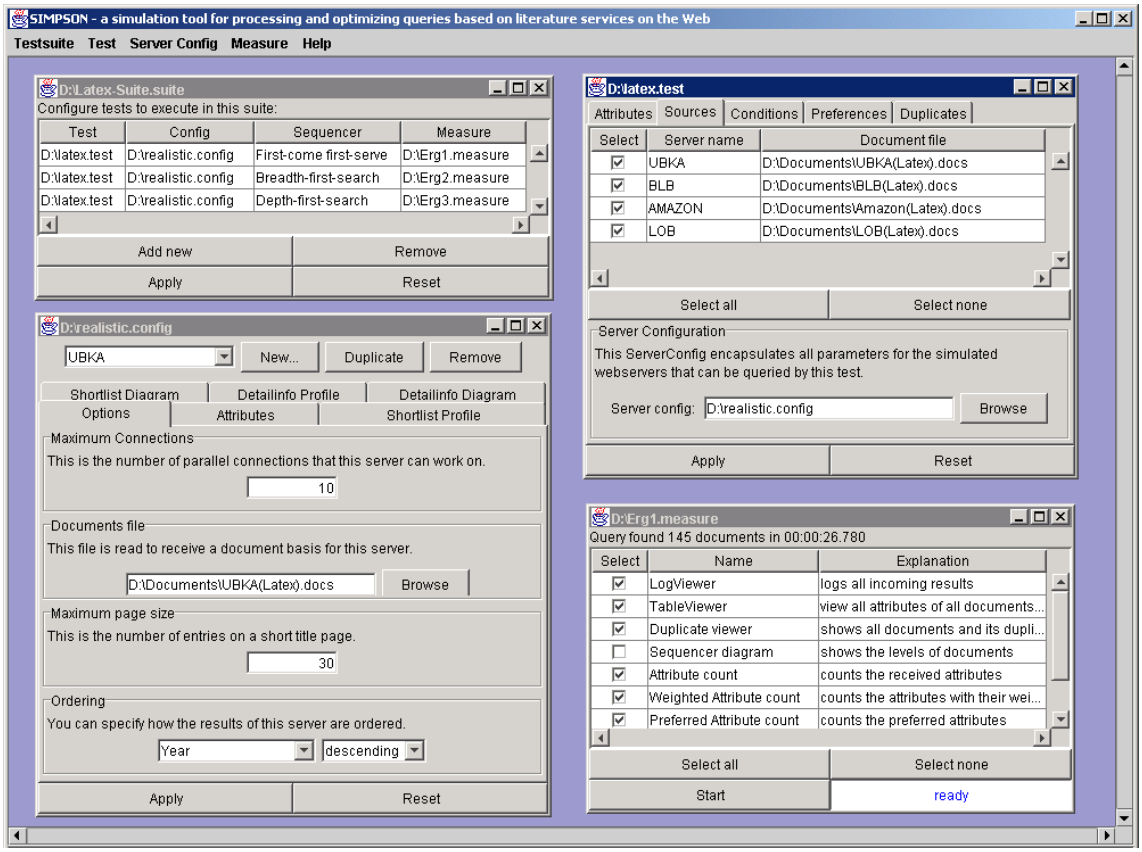


Abbildung 12.2: Benutzungsschnittstelle von SIMPSON

12.3 Testsuite

Eine Testsuite erlaubt es, eine Reihe von Tests in automatisierter Form durchzuführen. In Abbildung 12.2 (links oben) ist die Bedienoberfläche einer Testsuite zu sehen. Im angegebenen Beispiel sollen drei Tests hintereinander ausgeführt werden. Ein Test enthält die Angaben zu einer Anfrage und einige weitere Steuerparameter (vgl. Abschnitt 12.4). Die Daten eines Tests sind jeweils in einer Datei abgespeichert (s. Abbildung 12.1, Eingabeparameter). Der Name dieser Datei wird in der ersten Spalte (Test) angegeben, um auf den gewünschten Test Bezug zu nehmen.

Weiterhin wird eine Dienstspezifikation der zu befragenden Dienste benötigt (in der Spalte Config). Eine Dienstspezifikation legt die strukturellen Eigenschaften und das Antwortzeitverhalten der verfügbaren Dienste fest (vgl. Abschnitt 12.5). Die Dienstspezifikation wird ebenfalls über eine Datei bezogen (s. Abbildung 12.1, Diensteigenschaften). Darüber hinaus kann die gewünschte Ausführungsstrategie für jeden Test gewählt werden (Sequencer). In der letzten Spalte (Measure) wird schließlich der Name

einer neuen Datei angegeben, in welcher dann jeweils die Daten der Ergebnisbewertung (s. Abbildung 12.1, Ergebnisse & Zeiten) abgespeichert werden.

12.4 Test

Ein Test gibt im Wesentlichen die zu untersuchende Anfrage an. Zur leichteren Bedienbarkeit muss eine Anfrage nicht in MLS-QL-Syntax angegeben werden. Die Bestandteile eine MLS-QL-Anfrage können über separate Karteikarten (TabbedPanes) eingegeben werden, auf welchen entsprechende Hilfestellungen angeboten werden.

Zunächst kann angegeben werden, welche Attribute im Ergebnis gewünscht werden (Attributes). Diese Angabe entspricht in der MLS-QL-Notation der SELECT-Klausel. Zur Unterstützung der Attributauswahl steht eine vollständige Liste der Attribute des Domänenmodells zur Verfügung. In dieser Liste können die Attribute nicht nur ausgewählt, sondern auch mit Gewichten versehen werden, welche zu einem späteren Zeitpunkt im Rahmen der Bewertungsmaße herangezogen werden. Für die Anfragebearbeitung besagt die Auswahl der Attribute lediglich, bis zu welcher Ebene die Informationen bei jedem Dienst beschafft werden müssen.

Die zu befragenden Dienste (FROM-Klausel) können auf der nächsten Karteikarte angegeben werden (Sources, vgl. Abbildung 12.2, rechts oben). Im unteren Bereich des Fensters wird eine Dienstspezifikation ausgewählt. Daraufhin erscheinen im oberen Bereich des Fensters die Dienste, die in der gewählten Dienstspezifikation zur Verfügung stehen. Von diesen Diensten können nun die gewünschten Dienste markiert werden. In Abbildung 12.2 wurden alle verfügbaren Dienste ausgewählt, zwei Bibliotheksdienste (UBKA und BLB) und zwei Buchhändler (AMAZON und LOB). In der letzten Spalte der Tabelle (Document file) wird bei jedem Dienst die Datei angegeben, die die entsprechenden Ergebnisdokumente enthält. Im gegebenen Beispiel wurde zuvor jeder reale Dienst mit der Anfrage Titel=Latex befragt. Die Ergebnisse wurden in den entsprechenden Dateien abgelegt.

Die beiden nächsten Karteikarten erlauben es, harte (Conditions) und weiche (Preferences) Bedingungen zu formulieren. Das entspricht den WHERE- und den PREFERRING-Klauseln. Bei der Angabe von Präferenzen kann zusätzlich vermerkt werden, ob bei der Anfrageplanung die Strategie der präferenzbasierten Anfragezerlegung verwendet werden soll. Darüber hinaus kann jede Präferenz gewichtet werden. Diese Gewichte fließen wiederum in die Beurteilung im Zusammenhang mit den Bewertungsmaßen mit ein.

Auf der letzten Karteikarte (DUPLICATES) werden die Optionen für die Erkennung von Duplikaten eingestellt. Das entspricht der DUPLICATES-Klausel. Zum einen können die Attribute ausgewählt werden, die zur Duplikaterkennung herangezogen werden sollen (vgl. Abbildung 12.3). Zum anderen kann für jedes Attribut angeben werden, mit welcher Art von Vergleichen die Duplikate bestimmt werden sollen. Bei numerischen Attributen kann nur ein

exakter Vergleich verwendet werden. Für alphanumerische Attribute stehen ein exakter Vergleich, ein Vergleich, der eine Zeichenkette auf das Enthaltensein in der anderen Zeichenkette testet, sowie drei tolerante Vergleiche zur Verfügung (vgl. Kapitel 10). Jeder tolerante Vergleich ist mit einer Prozentangabe versehen, die den Grad der Toleranz widerspiegelt. Die Prozentangaben dienen als anschauliches Maß für die Übereinstimmung der beiden Zeichenketten. Wie dazu die entsprechenden Schwellwerte der Algorithmen berechnet werden, kann in [Obe02] nachgelesen werden. Die Duplikaterkennung erfolgt bei dem in Abbildung 12.3 dargestellten Beispiel lediglich aufgrund derselben ISBN.

Weiterhin kann angegeben werden, wie mit den erkannten Duplikaten in der Ergebnismenge verfahren werden soll. Dabei wird in diesem Fall von jeder Duplikatgruppe ein Stellvertreter ermittelt. Wie dieser berechnet wird, kann in der letzten Spalte (Merge action) eingestellt werden.

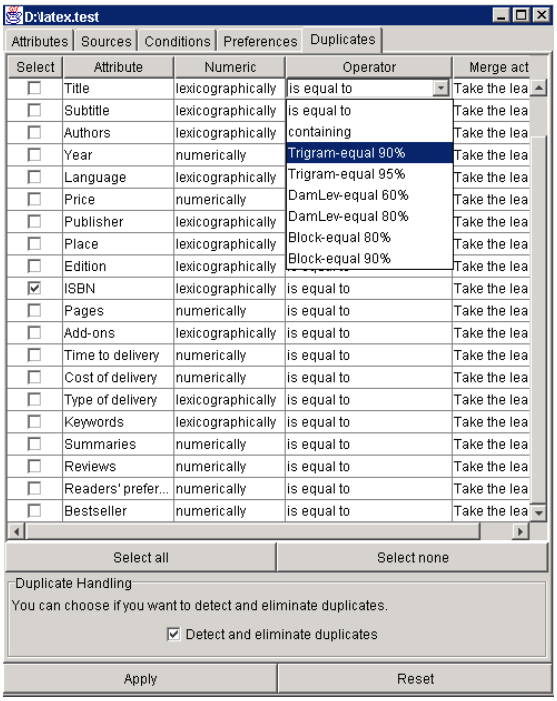


Abbildung 12.3: Spezifikation der Duplikaterkennung

Um einen Test auszuführen, muss im Test-Menü der Simulationsumgebung die Option Execute Test ausgewählt werden. Dabei erscheint ein zusätzliches Auswahlfenster zur Festlegung der gewünschten Ausführungsreihenfolge. Dabei stehen die in Abschnitt 9.5 erarbeiteten Strategien (FCFS, Breiten- und Tiefensuche) zur Verfügung.

12.5 Dienstspezifikation

Für die Spezifikation der strukturellen Eigenschaften und des Antwortzeitverhaltens der zu simulierenden Dienste steht ebenfalls ein Fenster mit einer Reihe von Karteikarten zur Verfügung.

Abbildung 12.4 zeigt die Spezifikation einer vertikalen Informationsportionierung (Attributes). Dazu werden die Attribute des Domänenmodells jeweils zusammen mit einer kurzen Beschreibung präsentiert. In der mittleren Spalte (Level) kann die Ebene der Informationsportion angegeben, auf der das Attribut erscheinen soll. In Abbildung 12.4 ist die Nachbildung eines Buchhandelskatalogs (AMAZON) zu sehen, bei dem bereits relativ viele Informationen (Titel, Autor, Jahr, Verlag, Preis, Leserbewertungen, Lieferzeit und Lieferkosten) auf der ersten Ebene geliefert werden. Insgesamt werden hier nur zwei Ebenen für die vollständige Lieferung der Informationen benötigt.

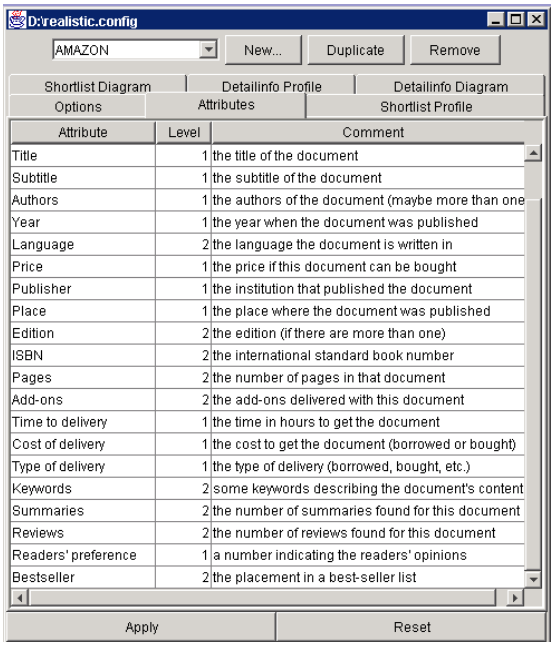


Abbildung 12.4: Spezifikation der vertikalen Informationsportionierung

Weiterhin kann das Antwortzeitverhalten eines Dienstes festgelegt werden. Dazu kann eine Antwortzeitverteilung für die Informationsportionen der ersten und aller folgenden Ebenen angegeben werden. Die Spezifikation einer Antwortzeitverteilung erfolgt über die Angabe der in Kapitel 8 eingeführten fünf Parameter (Abbildung 12.5, links). Gleichzeitig wird die Antwortzeitverteilung in graphischer Form dargestellt (Abbildung 12.5, rechts). Im angegebenen Beispiel wird die Antwortzeit des Bibliotheksdienstes UBKA gemäß der gefundenen Werte nachgebildet (vgl. Abschnitt 8.4).

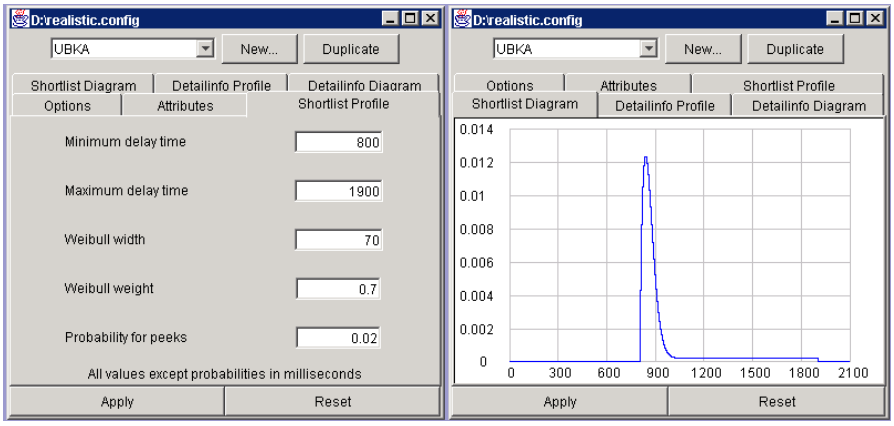


Abbildung 12.5: Spezifikation eines Antwortzeitprofils

In Abbildung 12.2 (links unten) ist die Konfiguration von weiteren Optionen zur Dienstspezifikation zu sehen (Options). An dieser Stelle wird die horizontale Informationsportionierung festgelegt sowie die Datei genannt, in welcher sich die entsprechenden Ergebnisdokumente befinden. Weiterhin kann angegeben werden, in welcher Sortierung der Dienst die gefundenen Dokumente zurückliefert und wie viele parallele Verbindungen bei der Anfrageausführung gleichzeitig benutzt werden dürfen.

In einer Dienstspezifikation können die Eigenschaften von mehreren Diensten gemeinsam festgelegt werden. In der Auswahlbox im oberen Bereich des Fensters sind alle Namen der verfügbaren Dienste aufgeführt. Zur Anzeige der Daten auf den Karteikarten wird jeweils der betreffende Dienstname dargestellt. In den Beispielen in diesem Kapitel wurden die strukturellen Eigenschaften von AMAZON sowie das Antwortzeitverhalten und die zusätzlichen Optionen von UBKA erläutert. Insgesamt umfasst die gegebene Dienstspezifikation vier Dienste (UBKA, BLB, AMAZON, LOB). In der vorliegenden Dienstspezifikation werden alle Dienste möglichst realitätsgetreu nachgebildet. Es ist allerdings auch möglich, eine Dienstspezifikation mit denselben Diensten, allerdings mit schnelleren Antwortzeiten oder mit einer größeren Anzahl an erlaubten parallelen Verbindungen zu erstellen. So muss zum Testen von einer Anfrageserie lediglich die Dienstspezifikation der betreffenden Dienste ausgetauscht werden, um das Verhalten der Anfrage bzw. der Anfragebearbeitung unter den veränderten Bedingungen zu ermitteln.

12.6 Ergebnisbewertung

Die Komponente der Ergebnisbewertung stellt eine Reihe von unterschiedlichen Anzeigeformen und Bewertungsmaßen für die kontinuierlich wachsende Ergebnismenge zur Verfügung. In Abbildung 12.2 (rechts unten) sind die angebotenen Darstellungsmittel zu sehen. Nachdem die gewünschten Darstellungsformen ausgewählt und angefordert (Start)

wurden, öffnen sich eine Reihe von neuen Fenstern, die den kontinuierlichen Aufbau der Ergebnismenge in der entsprechenden Form widerspiegeln (vgl. Abbildung 12.6).

Gespeichert werden von der Ergebnisbewertung die Ergebnisse mitsamt dem Zeitpunkt ihres Eintreffens in der Ergebnismenge (Ergebnisse & Zeiten). Im Gegensatz zu den anderen Dateien, die die Daten bzw. Einstellungen der Komponenten im XML-Format speichern (Eingabeparameter, Diensteigenschaften und Ergebnisdokumente), können die Daten der Ergebnisbewertung nur in Verbindung mit der Simulationsumgebung in übersichtlicher Form angezeigt und ausgewertet werden.

Die Protokollsicht (LogViewer) stellt das Eintreffen der einzelnen Teilergebnisse in textueller Form dar. Die Tabellensicht (TableViewer) stellt die Ergebnismenge in Form von einer Tabelle dar, deren Spalten den Attributen des Domänenmodells entsprechen. Die Duplikatsicht (Duplicate viewer) präsentiert eine Baumdarstellung der Ergebnismenge, wobei Duplikate jeweils unter demselben Knoten einsortiert werden. Zusätzlich zur Duplikatsicht kann ein Fenster angefordert werden, welches die erkannten Duplikate nochmals zahlenmäßig analysiert. In Abbildung 12.6 ist dieses Fenster aus Platzgründen nicht vollständig zu sehen. Folgende Tabelle 12.1 gibt eine Übersicht über die gefundenen Dokumente und die erkannten Duplikate:

LATEX	UBKA	BLB	AMAZON	LOB	Insgesamt
Dokumente	46	28	36	35	145
Duplikate	28	18	28	26	100
Duplikatgruppen					41

Tabelle 12.1: Übersicht über die gefundenen Dokumente und Duplikate

Insgesamt wurden zu der Anfrage nach Büchern, die im Titel das Wort Latex enthalten, 145 Dokumente gefunden. 100 dieser Dokumente wurden als Duplikate ausgewiesen. Insgesamt sind 41 Duplikatgruppen entstanden, d.h. ein Duplikat wurde durchschnittlich von 2,4 (100/41) Diensten gefunden. Damit wurden 45 Dokumente nur von einem der Dienste nachgewiesen.

Das Reihenfolgeprotokoll (Sequencer diagram, in Abbildung 12.6 nicht zu sehen) visualisiert die gerade bezogenen Informationsebenen der Dienste. Dadurch können die unterschiedlichen Ausführungsstrategien direkt verfolgt werden. Die folgenden drei Darstellungsformen präsentieren die in Kapitel 11 entwickelten Bewertungsmaße M2, M3 und M6. Für jedes Bewertungsmaß wird die Kurve zusammen mit dem berechneten Summenwert dargestellt. Durch die unterschiedlichen Farben[13] der Kurven wird deutlich, von welchem befragten Dienst die jeweils gerade gelieferte Informationsportion stammt. Zur Beurteilung der

[13] Für die farbige Darstellung der Abbildungen sei auf die elektronische Fassung der vorliegenden Arbeit verwiesen. Sie ist im elektronischen Volltextarchiv (EVA) der Universitätsbibliothek Karlsruhe zu finden, s. http://www.ubka.uni-karlsruhe.de/eva/index.html

Anfragebearbeitungsstrategien können die eintreffenden Attribute berücksichtigt werden (Attribute count, M2), die gewichteten Attribute (Weighted Attribute count, M3) oder auch die bevorzugten Werte zusammen mit den gewichteten Attributen (Preferenced Attribute count, M6). In Abbildung 12.6 enden die graphischen Darstellungen jeweils zu dem Zeitpunkt, an dem die Ergebnismenge vollständig vorhanden ist. Im durchgeführten Beispiel ist das nach 26,78 Sekunden.

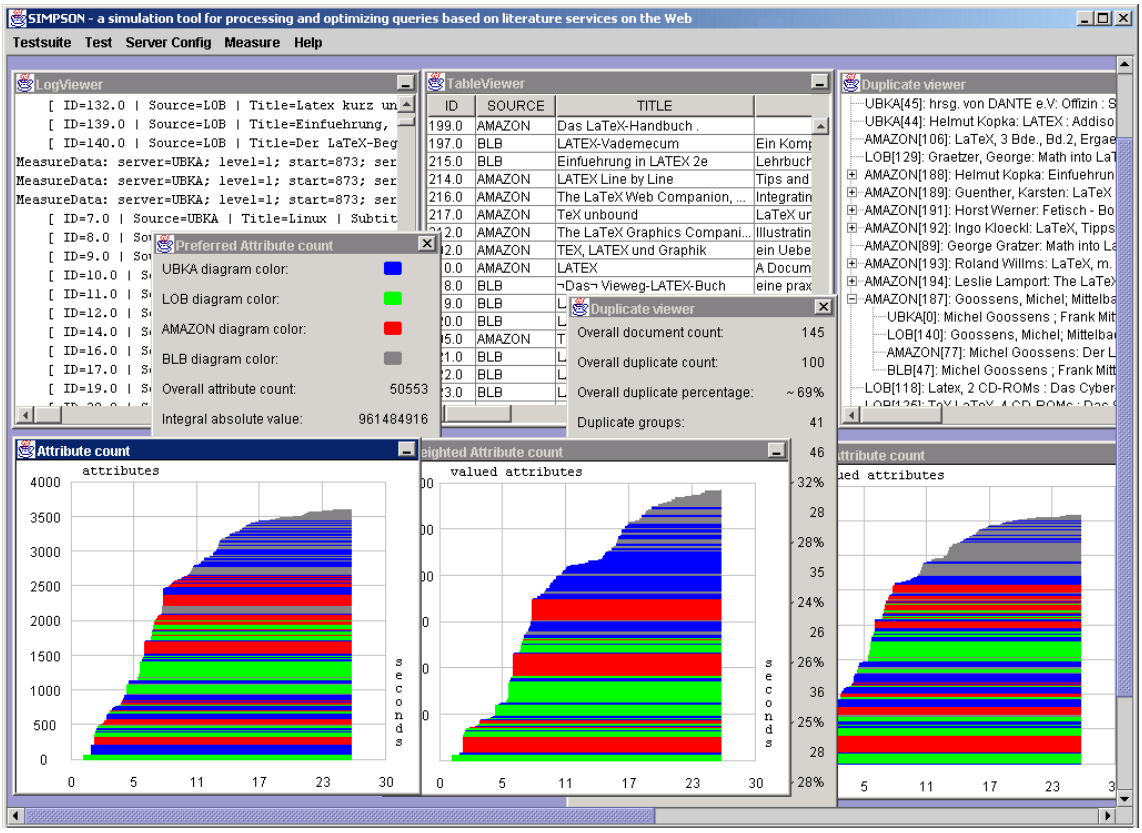


Abbildung 12.6: Darstellungsformen für die Ergebnisbewertung

12.7 Resümee

In diesem Kapitel wurde eine Simulationsumgebung namens SIMPSON entwickelt. Das Ziel der Simulationsumgebung und der damit durchführbaren Experimente wird es sein herauszufinden, welche Strategien der Anfragebearbeitung am besten geeignet sind bzw. in welchen Situationen und unter welchen Umständen welche Strategien der Anfragebearbeitung am besten geeignet sind.

Zunächst wurde die Grobarchitektur von SIMPSON entworfen. Die Grobarchitektur beinhaltet die drei Komponenten des Meta-Recherchesystems, die am Vorgang der

Anfragebearbeitung maßgeblich beteiligt sind. Die Komponenten beinhalten jeweils die in den Kapiteln 9 und 10 erarbeiteten Strategien zur Anfragebearbeitung, deren Wirksamkeit im Folgenden durch Experimente bewertet werden soll. Weiterhin stellt SIMPSON eine komfortable Benutzungsoberfläche zur Verfügung, mit welcher zahlreiche Optionen, Parameter und Strategien der Anfragebearbeitung in flexibler Weise ausgewählt bzw. variiert werden können. Mit SIMPSON können die zu befragenden Dienste in ihren strukturellen Eigenschaften und in ihrem Antwortzeitverhalten simuliert werden, so dass damit die vorhandenen Anfragestrategien nicht nur in real existierenden, sondern auch in beliebigen Szenarien untersucht werden können. Im folgenden Kapitel erfolgt nun die Durchführung dieser Experimente.

13 Experimente und Ergebnisse

13.1 Überblick

Im vorangegangenen Kapitel wurde eine Simulationsumgebung (SIMPSON) entwickelt, die die Durchführung von Experimenten zur Validierung der Anfragebearbeitungsstrategien in geeigneter Weise unterstützt. In diesem Kapitel erfolgt nun die Konzeption und die Durchführung von entsprechenden Experimenten.

Zunächst wird ein Grundszenario eingeführt, das die Basis für die folgenden Experimente bildet (Abschnitt 13.2). Das Grundszenario besteht aus drei Anfragen unterschiedlicher Größe und bildet die Literaturdienste gemäß ihrer aktuellen Eigenschaften nach. Zur Validierung der unterschiedlichen Anfragebearbeitungsstrategien werden die drei Komponenten der Anfragebearbeitung jeweils separat betrachtet. Ausgehend vom Grundszenario werden in Experimenten die unterschiedlichen Strategien der betreffenden Komponente untersucht und miteinander verglichen. Experiment 1 betrifft die Komponente der Ergebnisintegration und beschäftigt sich mit unterschiedlichen Strategien zur Duplikaterkennung (Abschnitt 13.3). In Experiment 2 werden die unterschiedlichen Strategien zur Reihenfolgebestimmung bei der Anfrageausführung untersucht (Abschnitt 13.4). Experiment 3 betrachtet die unterschiedlichen Strategien der Anfrageplanung (Abschnitt 13.5). Im vierten Experiment werden schließlich die Strategien der Anfragebearbeitung in zukünftigen Szenarien erprobt (Abschnitt 13.6).

Abschnitt 13.7 fasst die Ergebnisse der Experimente zusammen und leitet daraus Handlungsempfehlungen ab, sowohl für das Design von Meta-Recherchesystemen als auch für die Realisierung von Literaturdiensten.

13.2 Konzeption der Experimente und Grundszenario

Ziel der Experimente ist es, die unterschiedlichen Strategien der Anfragebearbeitung bewerten und miteinander vergleichen zu können. Dazu sollen die in Kapitel 9 und 10 erarbeiteten Strategien der drei Komponenten der Anfragebearbeitung untersucht werden.

Die im vorangegangenen Kapitel entwickelte Simulationsumgebung SIMPSON bietet die Möglichkeit, zahlreiche Optionen, Parameter und Einstellungen in flexibler Weise spezifizieren und variieren zu können. Um die Einflüsse der einzelnen Anfragestrategien genau identifizieren und anschließend miteinander vergleichen zu können, sollen möglichst viele der verfügbaren Optionen, Parameter und Einstellungen im Rahmen eines Experiments sowie zwischen den verschiedenen Experimenten konstant belassen werden.

Daher wird zunächst ein typisches Grundszenario festgelegt, das als Basis für die folgenden Experimente dienen soll. In diesem Szenario werden die betrachteten Dienste in realistischer Weise nachgebildet, d.h. mit derselben Informationsportionierung und demselben Antwortzeitverhalten, wie sie bzw. es bei den Untersuchungen in Kapitel 8 beobachtet werden konnte.

Im vorangegangenen Kapitel wurde bereits eine Dienstspezifikation eingeführt, die als typisch gelten kann. Sie enthält zwei Bibliotheksdienste (UBKA und BLB) und zwei Buchhandelskataloge (AMAZON und LOB). Im Rahmen einer Meta-Rechercheanfrage können sich diese beiden Dienstarten gut ergänzen: Bibliotheksdienste haben für einen Benutzer den Vorteil, dass sie eine kostenlose Beschaffung von Dokumenten anbieten. Buchhandelsdienste hingegen haben für einen Benutzer den Vorteil, dass sie eine Reihe von Zusatzinformationen zu den Dokumenten anbieten und die Dokumente teilweise schneller liefern können als Bibliotheksdienste, wenn auch zu einem deutlich höheren Preis. Da beide Dienstarten hauptsächlich Bücher in ihrem Bestand führen, ist diese Dienstkombination ebenso geeignet, um die Strategien der Duplikaterkennung testen zu können. Tabelle 13.1 gibt eine Übersicht über die gewählte Dienstspezifikation für das Grundszenario. Das Antwortzeitverhalten wird jeweils mit den in Abschnitt 8.4 eingeführten fünf Parametern beschrieben. Die ersten beiden Parameter geben den Bereich der regulären Antwortzeiten an. In diesem regulären Bereich kann ein charakteristischer Kurvenverlauf festgestellt werden, der im linken Teil eine Spitze aufweist und im rechten Teil daran anschließend eine Stufe. Der dritte Parameter gibt die Breite der Spitze, der vierte Parameter den Anteil der Spitze am gesamten Flächeninhalt unter der Kurve an. Bei der BLB liegen beispielsweise 80 % der Antwortzeiten im Bereich der Spitze und damit 20 % der Antwortzeiten im Bereich der anschließenden Stufe. Der fünfte Parameter gibt die Wahrscheinlichkeit von Ausreißern an.

Dienst	UBKA	BLB	AMAZON	LOB
Vert. Informationsportionierung, # Informationsebenen	3	3	2	2
Hor. Informationsportionierung, # Treffer pro Kurztitelanzeige	30	30	100	50
Zugelassener Parallelitätsgrad	10	10	25	25
Antwortzeitverhalten der 1. Informationsebene	800; 1900; 70; 0,7; 0,02	2500; 5000; 500; 0,8; 0,02	3000; 10000; 3000; 0,8; 0,02	1000; 3000; 200; 0,8; 0,02
Antwortzeitverhalten der 2. und 3. Informationsebene	1500; 3500; 100; 0,8; 0,02	2500; 4500; 200; 0,8; 0,02	500; 3000; 500; 0,8; 0,02	800; 2000; 100; 0,8; 0,02
Sortierung	Jahr desc	Jahr desc	Jahr desc	Jahr desc

Tabelle 13.1: Grundszenario für die Experimente (Diensteigenschaften)

Das Grundszenario soll weiterhin mehrere Anfragen mit unterschiedlich großen Ergebnismengen enthalten, um damit auch die Skalierbarkeit der Anfragestrategien untersuchen zu können. Als Anfragen für das Grundszenario wählen wir die bereits im vorangegangenen

Kapitel eingeführte Anfrage nach Büchern, deren Titel das Wort Latex enthält. Als Ergänzung dazu soll das Grundszenario auch eine größere Anfrage enthalten, dazu wählen wir die ebenfalls aus einigen Beispielen bekannte Anfrage nach Büchern, die Java in ihrem Titel enthalten. Tabelle 13.2 gibt eine Übersicht über die Anzahl der Dokumente, die die entsprechenden Dienste zu den jeweiligen Anfragen liefern. Um auch eine Anfrage mittlerer Größe zur Verfügung zu haben, wird die Java-Anfrage auf alle deutschsprachigen Dokumente eingeschränkt (Java-de). Die Ergebnismengen wurden in einem separaten Extraktionsprozess von den existierenden Diensten abgerufen, gemäß der Vorgaben des Domänenmodells transformiert und in SIMPSON verfügbar gemacht. Die Ergebnismengen spiegeln den Dokumentenbestand der Dienste vom Juni 2002 wider.

Dienst	UBKA	BLB	AMAZON	LOB	Insgesamt
Latex	46	28	36	35	145
Java-de	171	126	204	151	625
Java	447	171	792	500	1910

Tabelle 13.2: Grundszenario für die Experimente (Anfragen und Anfragegrößen)

Ausgehend von diesem Grundszenario sollen die folgenden Experimente zunächst jeweils eine Komponente der Anfragebearbeitung genauer untersuchen. Im ersten Experiment werden die unterschiedlichen Strategien zur Duplikaterkennung untersucht (Abschnitt 13.3). Experiment 2 betrachtet die unterschiedlichen Strategien zur Reihenfolgebestimmung bei der Anfrageausführung (Abschnitt 13.4). Und Experiment 3 testet die unterschiedlichen Strategien der Anfrageplanung (Abschnitt 13.5). Anschließend werden im vierten Experiment die Strategien der Anfragebearbeitung in zukünftigen Szenarien erprobt (Abschnitt 13.6).

13.3 Experiment 1: Strategien der Duplikaterkennung

Das erste Experiment soll die unterschiedlichen Strategien der Duplikaterkennung genauer untersuchen und validieren. Dazu werden zunächst Strategien betrachtet, die zur Bestimmung der Duplikate lediglich exakte Vergleiche einsetzen. Anschließend werden Strategien betrachtet, die mit toleranten Zeichenkettenvergleichen arbeiten. Die Duplikaterkennungsstrategien sollen sowohl in Bezug auf die Qualität ihrer Erkennungsleistung als auch im Hinblick auf den von ihnen zusätzlich verursachten Zeitaufwand untersucht werden.

13.3.1 Exakte Duplikaterkennung

Qualität. Zunächst soll die Qualität der exakten Duplikaterkennung betrachtet werden. In Abschnitt 10.2 wurde bereits argumentiert, dass aufgrund zahlreicher Tippfehler und Aufnahmeunterschiede in Literaturkatalogen exakte Vergleiche nicht alle tatsächlich vorhandenen Duplikate identifizieren können. In Kapitel 12 wurde bereits die Latex-Anfrage

exemplarisch durchgeführt und im Hinblick auf die identifizierten Duplikate ausgewertet. Zur Bestimmung der Duplikate wurden dabei die Dokumente hinsichtlich des ISBN-Attributs verglichen. Duplikate können allerdings auch durch andere Attributvergleiche gefunden werden. Beispielsweise können die Titel und die Autoren der Bücher miteinander verglichen werden. Da für neue Auflagen eines Buches auch neue ISBN-Angaben verwendet werden (vgl. ebenfalls Abschnitt 10.2), können mit dieser Strategie prinzipiell mehr Duplikate identifiziert werden als mit dem Vergleich der ISBN-Attribute. Dieselbe Semantik – und damit auch dieselbe Anzahl an Duplikaten – wie der Vergleich der ISBN-Angaben kann auch durch den Vergleich der Titel und der Jahreszahlen der Bücher erreicht werden.

Tabelle 13.3 zeigt die Anzahl der von den drei Strategien (ISBN, Titel+Autor, Titel+Jahr) identifizierten Duplikate bei der Latex-Anfrage. Zum Vergleich wird zusätzlich in der ersten Spalte (mit der Überschrift keine) die Anzahl der insgesamt von jedem Dienst gefundenen Dokumente angegeben (in Klammern, da es sich eben nicht um die Anzahl von Duplikaten handelt, wie in den anderen Spalten). In den Spalten ISBN, Titel+Autor und Titel+Jahr wird jeweils die Anzahl der Duplikate angegeben, die bei jedem Dienst identifiziert werden konnten. In der Zeile Summe Dupl. werden die insgesamt gefundenen Duplikate angegeben, in der Zeile Duplikatgruppen wird die Anzahl der gebildeten Duplikatgruppen ausgewiesen. Das Verhältnis Summe Dupl. zu Duplikatgruppen gibt an, wie viele Dokumente durchschnittlich in einer Duplikatgruppe enthalten sind und damit von wie vielen Diensten ein Duplikat durchschnittlich nachgewiesen wird. Setzt man die Duplikatzahl mit der Zahl der insgesamt gefundenen Dokumente ins Verhältnis, kann der Duplikatanteil innerhalb der Ergebnismenge bestimmt werden.

Anfrage	Latex				Java-de		Java	
Duplikatstrategie	keine	ISBN	Titel+Jahr	Titel+Autor	keine	ISBN	keine	ISBN
UBKA-Dupl.	(46)	28	18	23	(171)	101	(447)	282
BLB-Dupl.	(28)	18	18	20	(126)	84	(171)	115
AMAZON-Dupl.	(36)	28	2	0	(204)	137	(792)	459
LOB-Dupl.	(35)	26	3	0	(151)	125	(500)	408
Summe Dupl.	(145)	100	41	43	(652)	447	(1910)	1264
Duplikatgruppen	0	41	20	16	0	182	0	527
t_{max} (Sekunden)	20,5	20,6	20,1	20,4	1:16,4	1:17,0	2:42,9	2:45,4
M2 (Einheit 10^4)	5034	5110	5064	4999	96542	95105	4364508	4046492

Tabelle 13.3: Exakte Duplikaterkennung

Die Strategie der ISBN-Vergleiche identifiziert von den 145 gefundenen Dokumenten 100 als Duplikate, die sich auf 41 Duplikatgruppen aufteilen. Die Strategie der Titel+Jahr-Vergleiche erkennt dagegen nur 41 Duplikate, die in 20 Gruppen untergebracht werden, obwohl sie von derselben Duplikatsemantik wie die ISBN-Vergleiche ausgeht. Aufgrund zahlreicher Tippfehler und unterschiedlicher Titelaufnahmen können mit exakten Vergleichen nur weniger als die Hälfte der vorhandenen Duplikate erkannt werden. Die Strategie der Titel+Autor-Vergleiche könnte aufgrund der verwendeten Duplikatsemantik prinzipiell wesentlich mehr Dokumente als Duplikate identifizieren als die anderen beiden Strategien. Mit 43 erkannten Duplikaten ist die Erkennungsleistung dieser Strategie allerdings ebenfalls wenig überzeugend.

Insgesamt konnte bei der Latex-Anfrage ein recht hoher Anteil an Duplikaten in der Ergebnismenge (100/145=0,69) nachgewiesen werden. Diesbezüglich wurden auch die anderen beiden Anfragen des Grundszenarios untersucht. Bei der Java-de-Anfrage ergab sich ebenfalls ein Duplikatanteil von 69 %, bei der Java-Anfrage wurde ein Duplikatanteil von 66 % ermittelt. Diese Werte unterstreichen noch einmal den Nutzen und die Synergieeffekte von Meta-Recherchesystemen. Etwa 2/3 der gefundenen Ergebnisse sind Duplikate, d.h. die Größe der Ergebnismenge kann durch eine Duplikaterkennung um mehr als 1/3 verringert werden. Im Latex-Beispiel werden nun nur noch 145-100+41=86 statt vorher 145 Dokumente angezeigt. Gleichzeitig wird die Ergebnismenge durch die Duplikaterkennung und -verschmelzung auch verdichtet, da zu den Duplikaten nun mehrere Beschaffungsalternativen und i.d.R. auch mehrere Zusatzinformationen vorhanden sind.

Zeitaufwand. Das Zeitverhalten einer Anfragebearbeitungsstrategie kann anhand zweier verschiedener Größen beurteilt werden: Zum einen an der Zeit t_{max}, die bis zum Eintreffen des letzten Teilergebnisses benötigt wird, und zum anderen am Summen- bzw. Integralwert eines der in Kapitel 11 eingeführten Bewertungsmaße. Da wir in diesem Experiment die Duplikaterkennung möglichst unabhängig von anderen Komponenten und damit auch von den Präferenzen des Benutzers betrachten wollen, wählen wir als Bewertungsmaß das reine Attributmaß M2 (s. Abschnitte 11.4.2 und 12.6). Der zusätzliche Zeitaufwand der exakten Duplikaterkennung kann nun dadurch ermittelt werden, dass die beiden Werte t_{max} und M2 der betreffenden Anfrage mit den entsprechenden Werten derselben Anfrage ohne Duplikat-erkennung verglichen werden.

In Tabelle 13.3 sind t_{max} und M2 für jede Anfrage und Duplikaterkennungsstrategie angegeben. Da die Antwortzeiten der Literaturdienste zufällig entsprechend der Wahrscheinlichkeitsverteilung des Antwortzeitprofils generiert werden, variieren die Werte von t_{max} und M2 leicht für unterschiedliche Durchläufe derselben Anfrage. Daher wurde jede Anfrage 5 Mal ausgeführt. Die in Tabelle 13.3 angegebenen Werte für t_{max} und M2 entsprechen dem Mittelwert der jeweiligen fünf Einzelwerte. M2 wurde auf der Basis von Millisekunden berechnet, d.h. die Anzahl der vorhandenen Attribute wurde pro Millisekunde ermittelt und aufsummiert. Um unterschiedliche Anfragen anhand der Werte von M2

vergleichen zu können, wurde die Summe jeweils bis zu einem identischen Zeitpunkt berechnet. Dieser Zeitpunkt wurde gleichzeitig bzw. später als das Eintreffen des letzten Ergebnisses der langsamsten Anfrage gewählt. Für die Latex-Anfrage wurde M2 beispielsweise jeweils über den Zeitraum von 25 Sekunden berechnet.

Bei der Latex-Anfrage hat der Einsatz einer Duplikaterkennungsstrategie keine charakteristischen Auswirkungen auf das beobachtete Antwortzeitverhalten. Die Schwankungen der Werte von t_{max} und M2 lassen sich mehr auf die Variationen der simulierten Antwortzeiten der befragten Dienste zurückzuführen als auf den Einsatz von Duplikaterkennungsstrategien. Bei größeren Ergebnismengen dagegen können bereits Verzögerungen beim Einsatz einer exakten Duplikaterkennung beobachtet werden, wenn auch nur in geringer Weise. Die vollständige Beantwortung einer Anfrage dauert mit einer Duplikaterkennung einige Sekunden länger, der berechnete Summen- bzw. Integralwert fällt etwas geringer aus, d.h. eine Reihe von Attributen treffen später in der Ergebnismenge ein.

Bei kleinen bis mittelgroßen Ergebnismengen erzeugt die exakte Duplikaterkennung also keine nennenswerten Verzögerungszeiten. Bei sehr großen Ergebnismengen (von mehr als 1000 Treffern), wie sie in der Praxis eher selten vorkommen (sollten), macht sich eine geringe Verzögerung durch die Duplikaterkennung bemerkbar, die allerdings angesichts der insgesamt benötigten Antwortzeit auch eher vernachlässigt werden kann. D.h. die Strategien zur exakten Duplikaterkennung skalieren auch bei großen Ergebnismengen sehr gut.

13.3.2 Tolerante Duplikaterkennung

Qualität. In weiteren Experimenten soll nun zunächst die Qualität der toleranten Duplikaterkennungsstrategien untersucht werden. Dazu wird wiederum die Duplikatsemantik der ISBN-Vergleiche nachgebildet, indem die Titel auf tolerante und die Jahreszahlen auf exakte Weise miteinander verglichen werden. Dabei werden die in Kapitel 10 entwickelten Algorithmen jeweils mit zwei unterschiedlichen Schwellwerten getestet (vgl. Abschnitt 12.4).

Tabelle 13.4 zeigt die Anzahl der von den toleranten Duplikaterkennungsstrategien gefundenen Duplikate. Im Vergleich zur exakten Duplikaterkennung (Tabelle 13.3) können deutlich mehr der vorhandenen Duplikate identifiziert werden. Allerdings können auch mit toleranten Zeichenkettenvergleichen nur etwa 2/3 der vorhandenen Duplikate gefunden werden. Die Erkennungsleistung der untersuchten Strategien liegt bei der Latex-Anfrage zwischen 46 % und 67 %, durchschnittlich beträgt sie 61 %.

In Bezug auf die Qualität der Erkennungsleistung wurden auch Experimente mit den anderen beiden Anfragen des Grundszenarios durchgeführt. Die Ergebnisse sind in Tabelle 13.5 und 13.6 zu sehen. Bei der Java-de-Anfrage variiert die Erkennungsleistung der getesteten Strategien zwischen 55 % und 85 %, durchschnittlich beträgt sie 67 %. In Bezug auf die Java-Anfrage können durchschnittliche Erkennungsraten von 72 % ermittelt werden, die im Bereich zwischen 58 % und 92 % schwanken.

Latex-Anfrage Duplikatstrategie	keine	ISBN	Tit(TRI95) +Jahr	Tit(TRI90) +Jahr	Tit(DL80) +Jahr	Tit(DL60) +Jahr	Tit(BL90) +Jahr	Tit(BL80) +Jahr
UBKA-Dupl.	(46)	28	24	26	24	24	17	23
BLB-Dupl.	(28)	18	18	18	18	19	15	16
AMAZON-Dupl.	(36)	28	7	9	8	10	7	11
LOB-Dupl.	(35)	26	12	14	11	13	7	12
Summe Dupl.	(145)	100	61	67	61	66	46	62
Duplikatgruppen	0	41	27	27	26	27	19	26
t_{max} (Sekunden)	20,5	20,6	21,2	21,4	20,9	20,4	20,7	20,7
M2 (Einheit 10^4)	5034	5110	4965	4950	4942	4998	4977	4969

Tabelle 13.4: Tolerante Duplikaterkennung (Latex)

Zeitaufwand. Nun soll der zusätzliche Zeitaufwand für die toleranten Duplikaterkennungsstrategien ermittelt werden. In Kapitel 10 wurde bereits die algorithmische Komplexität dieser Verfahren bemerkt. In den Experimenten sollen nun die Auswirkungen und Konsequenzen für die praktische Einsetzbarkeit herausgefunden werden.

Bei den Antwortzeiten der Latex-Anfrage können bereits kleine Verzögerungen durch den Einsatz von toleranten Zeichenkettenvergleichen beobachtet werden. Auch wenn t_{max} teilweise noch unter dem Referenzwert (der Anfrage ohne Duplikaterkennung) liegt, sind die Summen- bzw. Integralwerte alle kleiner als der Referenzwert. Die beobachteten Verzögerungen sind allerdings nicht signifikant und können vernachlässigt werden.

Java-de-Anfrage Duplikatstrategie	keine	ISBN	Tit(TRI95) +Jahr	Tit(TRI90) +Jahr	Tit(DL80) +Jahr	Tit(DL60) +Jahr	Tit(BL90) +Jahr	Tit(BL80) +Jahr
UBKA-Dupl.	(171)	101	87	99	90	108	74	87
BLB-Dupl.	(126)	84	67	70	70	74	59	65
AMAZON-Dupl.	(204)	137	28	40	33	60	61	119
LOB-Dupl.	(151)	125	65	86	67	112	75	111
Summe Dupl.	(652)	447	247	295	260	354	269	382
Duplikatgruppen	0	182	87	97	96	114	93	76
t_{max} (Min:Sek)	1:16,4	1:17,0	1:19,1	1:23,4	1:17,8	1:29,9	1:26,5	9:52,6
M2 (Einheit 10^4)	96542	95105	88241	81106	89401	74548	82120	—

Tabelle 13.5: Tolerante Duplikaterkennung (Java-de)

Bei der Java-de-Anfrage sind die Verzögerungen bereits deutlicher zu sehen. Der t_{max}-Wert verlängert sich gerade bei den Vergleichen mit einer höheren Toleranz in erheblichem Maße. Ursache dafür ist die Implementierung der Vergleiche: Bei Vergleichen mit einer geringeren Toleranz, d.h. mit einem niedrigeren Schwellwert, können die Algorithmen die Berechnungen

früher beenden, nämlich sobald klar ist, dass der Schwellwert überschritten wird. Gerade für die Damerau-Levenshtein-Metrik existiert dazu eine sehr effiziente Implementierung, die auch in der vorliegenden Arbeit verwendet wird (s. Abschnitt 10.5). Für die TRI95- und die DL80-Strategie können die zusätzlichen Verzögerungen bei der vorliegenden Anfragegröße noch vernachlässigt werden. Der Zeitaufwand für den in dieser Arbeit neu entwickelten Block-Algorithmus explodiert allerdings bei einer Ergebnismengengröße von 652 Dokumenten und einer Toleranz von 80 %. Qualitativ kann der Block-Algorithmus zwar eine sehr gute Erkennungsleistung aufweisen, allerdings existiert bisher keine annähernd so effiziente Implementierung wie für die Trigramm- und die Damerau-Levenshtein-Vergleiche.

Bei der größten der drei Anfragen, der Java-Anfrage, können schließlich die bisher beobachteten Tendenzen manifestiert werden. Die effizienteste Implementierung steht zweifelsfrei für die Damerau-Levenshtein-Vergleiche mit einer relativ geringen Toleranz zur Verfügung. Allerdings führt auch das DL80-Verfahren spätestens bei Ergebnismengengrößen von 1910 Dokumenten zu drastischen Verzögerungen.

Java-Anfrage Duplikatstrategie	keine	ISBN	Tit(TRI95) +Jahr	Tit(TRI90) +Jahr	Tit(DL80) +Jahr	Tit(DL60) +Jahr	Tit(BL90) +Jahr	Tit(BL80) +Jahr
UBKA-Dupl.	(447)	282	238	277	233	305	216	269
BLB-Dupl.	(171)	115	98	101	99	105	79	91
AMAZON-Dupl.	(792)	459	202	272	193	315	249	457
LOB-Dupl.	(500)	408	221	300	213	360	240	349
Summe Dupl.	(1910)	1264	759	950	738	1085	784	1166
Duplikatgruppen	0	527	280	310	287	335	274	225
t_{max} (Min:Sek)	2:42,9	2:45,4	3:32,2	4:45,3	3:17,5	5:17,2	3:35,2	1:05:19,1

Tabelle 13.6: Tolerante Duplikaterkennung (Java)

Für mittelgroße Ergebnismengen (bis etwa 500 Treffer) ist es ohne merklichen Zusatzaufwand möglich, tolerante Duplikaterkennungsstrategien einzusetzen. Dabei sind hinsichtlich des zusätzlichen Zeitaufwands Strategien mit einer niedrigeren Toleranz zu bevorzugen, auch wenn diese dadurch etwas weniger Duplikate identifizieren können.

Betrachtet man die Experimente mit den exakten und den toleranten Duplikaterkennungsstrategien, so hat sich die Strategie der ISBN-Vergleiche – sowohl was die Erkennungsleistung betrifft als auch was eine effiziente Implementierbarkeit angeht – als überlegen erwiesen. Der Nachteil dieser Strategie besteht allerdings darin, dass sie von Meta-Recherchesystemen, die die Informationsportionen der zweiten Ebene nicht automatisch beschaffen (und das trifft bisher auf alle existierenden Ansätze zu), nicht verwendet werden kann.

13.4 Experiment 2: Strategien der Anfrageausführung

In diesem Experiment sollen die unterschiedlichen Strategien der Anfrageausführung untersucht werden. Die Strategien unterscheiden sich in der Reihenfolge, in der die zur Beantwortung einer Anfrage erforderlichen Informationsportionen beschafft werden. Es stehen drei Strategien zur Verfügung: Die Strategie der Breitensuche, die zuerst alle Kurztitelinformationen von den gefundenen Dokumenten beschafft, die Strategie der Tiefensuche, die von jedem gefundenen Dokument sofort alle verfügbaren Informationen beschafft, und die Strategie First Come First Serve (FCFS), die die Informationsportionen in der Reihenfolge, in der sie verfügbar geworden sind, beschafft (vgl. Abschnitt 9.5). Die FCFS-Strategie stellt somit eine Mischung zwischen der Breiten- und der Tiefensuche dar. Sie zeichnet sich dadurch aus, dass sie gerade zu den ersten Dokumenten in der Ergebnismenge auch schnell sämtliche Zusatzinformationen beschafft.

Tabelle 13.7 zeigt die Ergebnisse eines Experiments, in welchem die Latex-Anfrage mit den unterschiedlichen Strategien der Anfrageausführung auf den Diensten des Grundszenarios durchgeführt wurde. Die Anfrage wurde mit jeder Ausführungsstrategie 5 Mal getestet. In Tabelle 13.7 sind die durchschnittlichen Antwortzeiten (t_{max}) zu sehen, die Durchschnittswerte von M2 wurden jeweils über einem Zeitraum von 25 Sekunden berechnet.

Reihenfolge	Breitensuche	Tiefensuche	FCFS
t_{max} (Sekunden)	20,2	20,3	19,6
M2 (Einheit 10^4)	5136	5069	5137

Tabelle 13.7: Strategien der Anfrageausführung (Latex)

Die Strategien der Anfrageausführung haben prinzipiell keine Auswirkungen auf t_{max}, da sie jeweils den bei jedem Dienst zur Verfügung stehenden Parallelitätsgrad vollständig ausnutzen, auch wenn dadurch die vorgegebene Reihenfolge der Strategie nicht mehr streng eingehalten werden kann. Die in Tabelle 13.7 beobachtbaren Schwankungen von t_{max} sind also auf die Varianzen der generierten Antwortzeiten zurückzuführen. Auf den Wert von M2 haben die Strategien der Anfrageausführung ebenfalls keinen signifikanten Einfluss, zumindest nicht im vorliegenden Grundszenario. Das bedeutet, dass die Attribute bei den betrachteten Diensten insgesamt relativ gleichmäßig auf den unterschiedlichen Informationsebenen verteilt sind.

Im folgenden Experiment wird die gegebene Latex-Anfrage nur an die zwei Bibliotheksdienste des Grundszenarios gestellt. Da die Bibliotheksdienste die Attribute über mehrere Ebenen verteilen und nur noch verhältnismäßig wenig Attribute auf der dritten Ebene angezeigt werden (nämlich die Beschaffungsoptionen), kann das unterschiedliche Verhalten der Ausführungsstrategien in diesem Szenario deutlicher wahrgenommen werden. Tabelle 13.8 zeigt die Ergebnisse dieses Experiments.

Reihenfolge	Breitensuche	Tiefensuche	FCFS
t_{max} (Sekunden)	20,2	20,1	20,1
M2 (Einheit 10^4)	2371	2275	2345
M3 (Einheit 10^4)	436	495	463

Tabelle 13.8: Strategien der Anfrageausführung bei Bibliotheksdiensten (Latex)

Zunächst kann beobachtet werden, dass die ermittelten Antwortzeiten (t_{max}) in etwa identisch sind mit den Ergebnissen des vorangegangenen Experiments (s. Tabelle 13.7), in welchem alle vier Dienste des Grundszenarios befragt wurden. Daraus kann geschlossen werden, dass die Anfragen an die Buchhandelsdienste insgesamt schneller beantwortet werden können als die Anfragen an die Bibliotheksdienste – sei es, weil die Buchhandelsdienste im Allgemeinen schnellere Antwortzeiten aufweisen oder weil bei ihnen weniger Informationsebenen berücksichtigt werden müssen oder weil sie weniger Dokumente zu der gegebenen Anfrage bereitstellen.

Bei den Bibliotheksdiensten gibt das Bewertungsmaß M2 bei der Strategie der Tiefensuche den geringsten Wert an. Das lässt sich dadurch erklären, dass bei der Tiefensuche im zeitlichen Verlauf der Anfragebearbeitung relativ früh Informationsportionen mit nur wenigen Attributen beschafft werden.

Ein anderes Ergebnis kann hingegen gefunden werden, wenn die Attribute individuell gewichtet werden, beispielsweise nach den Vorlieben des Benutzers. Gewichtet man z.B. die Attribute, die die Lieferbedingungen angeben, um den Faktor 10 höher als die übrigen Attribute des Domänenmodells (Lieferbedingungen mit Gewicht 1, die übrigen Attribute mit Gewicht 0,1), ergibt sich mit dem Bewertungsmaß M3 ein deutlich größerer Wert für die Strategie der Tiefensuche im Verhältnis zu den anderen beiden Strategien. D.h. in einem derartigen Fall ist die Strategie der Tiefensuche den anderen beiden Strategien überlegen und vorzuziehen, da sie die vom Benutzer gewünschten Attribute wesentlich schneller liefern kann.

Werden im Grundszenario die Bibliothekskataloge ohne eine zusätzliche Gewichtung der Attribute befragt, erreicht man mit der Strategie der Breitensuche im Verhältnis zu den anderen beiden Strategien das beste Antwortzeitverhalten (vgl. M2 in Tabelle 13.8). Im folgenden Experiment sollen nun die Antwortzeiten des Grundszenarios, genauer gesagt die der Bibliotheksdienste, variiert werden, um herauszufinden, ob die Antwortzeiten der Dienste ebenfalls Auswirkungen auf die Eignung der Ausführungsstrategien haben. Dazu werden die Antwortzeiten der zweiten und dritten Informationsebene jeweils um 1 Sekunde beschleunigt, d.h. die beiden Parameter MinZeit und MaxZeit werden um 1 Sekunde herabgesetzt. In Tabelle 13.9 ist das Ergebnis dieses Experiments zu sehen.

Reihenfolge	Breitensuche	Tiefensuche	FCFS
t_{max} (Sekunden)	14,3	14,1	14,3
M2 (Einheit 10^4)	2633	2683	2670

Tabelle 13.9: Strategien der Anfrageausführung bei schnelleren Bibliotheksdiensten (Latex)

Insgesamt ist die Antwortzeit (t_{max}) dadurch um ca. 5 Sekunden schneller geworden (vgl. t_{max} in Tabelle 13.8). Betrachtet man die Werte von M2, so schneidet in diesem Experiment die Strategie der Tiefensuche besser als die anderen beiden Strategien ab. Um die Werte von M2 mit dem vorangegangenen Experiment (s. Tabelle 13.8) vergleichen zu können, wurden sie in diesem Experiment ebenfalls über den Zeitraum von 25 Sekunden berechnet. Die Werte von M2 sind in diesem Experiment insgesamt größer, da die Ergebnisse ja durch das geänderte Antwortzeitverhalten der Dienste schneller geliefert werden können.

Die Experimente dieses Abschnitts zeigen, dass es nicht möglich ist, die unterschiedlichen Strategien der Anfrageausführung im Allgemeinen zu bewerten. Die Antwortzeit t_{max} ist aufgrund des vollständig ausgenutzten Parallelitätsgrades für alle Strategien im Wesentlichen identisch. Unterschiede treten nur bei der Reihenfolge der eintreffenden Informationsportionen auf, d.h. die Unterschiede lassen sich beispielsweise über die Bewertungsmaße M2 und M3 verdeutlichen. Für alle Strategien sind Szenarien denkbar, in denen eine Strategie den anderen beiden Strategien jeweils überlegen ist. In den durchgeführten Experimenten wurde deutlich, dass für die Bewertung und Eignung der untersuchten Strategien mehrere Faktoren von Bedeutung sind: zum einen die vertikale Informationsportionierung der befragten Dienste, zum anderen eine gegebene Gewichtung der Attribute und schließlich auch die Antwortzeiten der befragten Dienste in Bezug auf die unterschiedlichen Informationsebenen.

13.5 Experiment 3: Strategien der Anfrageplanung

Dieses Experiment beschäftigt sich mit der Anfrageplanung. In Abschnitt 9.4 wurde die Strategie der präferenzbasierten Anfragezerlegung entwickelt. In diesem Experiment soll nun der Mehrwert dieser Strategie gegenüber der klassischen Anfrageplanung (welche die gegebene Anfrage sozusagen ohne Änderungen an die Dienste weiterreicht) untersucht werden. Die präferenzbasierte Anfragezerlegung kann nur erfolgen, wenn die gegebene Anfrage mindestens eine Präferenz enthält.

Im Laufe der Arbeit (z.B. in Abschnitt 6.6 oder 7.4) sind bereits etliche Beispiele für Präferenzen gegeben worden. Für die Formulierung von Präferenzen eignen sich die Attribute Sprache, Jahr und Lieferzeit besonders gut. Im Folgenden wollen wir nun zunächst die Verteilung der Werte bei diesen Attributen untersuchen und explizit angeben. Dazu wählen wir die größte Anfrage des Grundszenarios, die Java-Anfrage. In Tabelle 13.10 ist angegeben, wie viele Dokumente der Gesamtanfrage den jeweiligen Wert unter dem genannten Attribut

aufweisen. Insgesamt liefert die Java-Anfrage 1910 Dokumente. Summiert man die Anzahl der Dokumente bei den unterschiedlichen Werten eines Attributs auf, so wird die Gesamtzahl der Dokumente (1910) jeweils nicht ganz erreicht. Das rührt daher, dass zum Teil nicht für jedes Dokument diese Attribute in den Katalogen vorhanden sind und zum Teil vorhandene Attributwerte auch nicht immer korrekt extrahiert werden konnten.

Jahr	# Dokumente
2002	136
2001	462
2000	374
1999	366
1998	218
1997	149
1996	101
1995	9
< 1995	62

Sprache	# Dokumente
deutsch	652
englisch	1174
andere	6

Lieferzeit	# Dokumente
<=2 Stunden	283
<=3 Tage	535
<=2 Wochen	562
>2 Wochen	159

Tabelle 13.10: Werteverteilungen bei ausgewählten Attributen

Die vorhandenen Dokumente weisen unter dem Attribut Sprache im Wesentlichen nur zwei unterschiedliche Werte auf, nämlich deutsch und englisch. Gibt der Benutzer beispielsweise eine derartige Bedingung an das Sprach-Attribut an, so kann die Größe der Ergebnismenge dadurch nahezu halbiert werden. Für die beiden anderen Attribute Lieferzeit und Jahr sind wesentlich mehr unterschiedliche Werte unter jedem Attribut vorhanden, so dass die Angabe eines einzelnen Wertes die Ergebnismenge zwar deutlich verkleinern könnte, in der Regel allerdings zu speziell ist (besonders was die Lieferzeit betrifft). Realistischer sind bei diesen Attributen eher Bereichsangaben, wie sie auch bereits bei der Übersicht über die Lieferzeiten verwendet wurden.

Im Folgenden soll nun ein Experiment durchgeführt werden, welches ermittelt, in welcher Zeit die vom Benutzer bevorzugten Dokumente in der Ergebnismenge vorhanden sind. Durch die präferenzbasierte Anfragezerlegung werden eben genau die bevorzugten Dokumente zuerst beschafft. In diesem Experiment werden die Präferenzen zunächst separat und anschließend in Kombinationen untersucht. Verglichen mit den in Abschnitt 6.6 oder 7.4 betrachteten Präferenzen werden die Präferenzen für dieses Experiment etwas vereinfacht. Die Präferenz bezüglich des Attributs Jahr bevorzugt Dokumente, die im Jahr 2000 oder später erschienen sind. Die Präferenz in Bezug auf Sprache betrifft Dokumente in deutscher Sprache. Und im Hinblick auf die Lieferzeit bevorzugt die Präferenz Dokumente, die in maximal drei Tagen geliefert werden können.

In Tabelle 13.11 ist zunächst t_{max} der Java-Anfrage angegeben. Diese Anfrage enthält keine Präferenzen. Der Wert t_{max} ist für alle Anfragen mit den jeweils angegebenen Präferenzen im Wesentlichen gleich. Entscheidender ist vielmehr der in der nächsten Zeile angegebene Wert t_{pref}. Der Wert von t_{pref} gibt für jede Anfrage mit der entsprechenden Präferenz den Zeitpunkt an, an dem alle bevorzugten Dokumente mit sämtlichen Zusatzinformationen in der Ergebnismenge eingetroffen sind. In der darunter liegenden Zeile ist jeweils die Anzahl der Dokumente angegeben, die die gegebene Präferenz bzw. Kombination der Präferenzen erfüllen. Die angegebenen Zeiten sind wiederum Durchschnittszeiten, die aus 5 Anfragedurchläufen ermittelt wurden.

Gibt der Benutzer zu seiner Java-Anfrage beispielsweise eine Präferenz an, so muss er trotz der großen Menge an Treffern nur etwa 1 Minute warten, bis all seine bevorzugten Dokumente mit vollständigen Informationen vorhanden sind. Gibt der Benutzer zusätzlich sogar zwei Präferenzen an, so muss er nur etwa eine halbe Minute auf die besten Ergebnisse warten. Bei der Angabe von drei Präferenzen können die bevorzugten Dokumente sogar in weniger als 10 Sekunden beschafft werden. Damit liegt der Vorteil der präferenzbasierten Anfragezerlegung auf der Hand. Auch bei großen Ergebnismengen können die für den Benutzer relevanten Ergebnisse in weniger als 10 Sekunden, was ja auch eine kritische Zeitmarke ist (s. Abschnitt 8.3.1.3), geliefert werden – eben unter der Voraussetzung, dass sich der Benutzer die Mühe macht und explizit einige Präferenzen angibt.

Präferenzen	keine	Jahr	Sprache	Lieferzeit	Jahr + Sprache	Jahr + Lieferzeit	Sprache + Lieferzeit	Jahr + Sprache + Lieferzeit
t_{max} (Min:Sek)	2:42,9							
t_{pref} (Min:Sek od. Sekunden)		53,9	1:15,8	1:15,2	21,4	15,0	33,9	8,9
# Dokumente	1910	972	652	818	320	405	278	190
M4 (Einheit 10^4)	26201	31345	31134	31466	32501	32822	32140	33532

Tabelle 13.11: Präferenzbasierte Anfragezerlegung

Bei der aktuellen Gestaltung der Bibliothekskataloge ist es nicht möglich, eine Anfrage zu formulieren, die eine Bedingung an die Lieferzeit enthält. Daher spiegelt das Experiment in diesem Punkt die Realität nicht exakt wider. Andererseits macht das Beispiel aber auch deutlich, wie groß der Gewinn für den Benutzer sein könnte, wenn Bibliothekskataloge in Zukunft gerade das Attribut Lieferzeit direkter unterstützen würden.

Bei der Betrachtung der Anzahl der Dokumente und t_{pref} fällt auf, dass dieses Verhältnis nicht proportional ist. Bei der Jahr+Sprache-Präferenz wird beispielsweise im Verhältnis zur Jahr+Lieferzeit-Präferenz eine längere Zeit benötigt, obwohl weniger Dokumente beschafft

werden müssen. Da die gefundenen Dokumente von unterschiedlichen Diensten geliefert werden und wir bereits im vorangegangenen Experiment 2 (Abschnitt 13.4) bemerkt haben, dass die Bibliotheksdienste insgesamt langsamer antworten, liegt die Vermutung nahe, dass bei der Jahr+Sprache-Präferenz ein Großteil der Dokumente von den Bibliotheksdiensten stammen.

Zur Verdeutlichung des Gewinns durch die Angabe von Präferenzen und durch den Einsatz der präferenzbasierten Anfragezerlegung werden in der letzten Zeile von Tabelle 13.11 die Werte für M4 angegeben (vgl. Abschnitt 11.4). Dieses Maß berücksichtigt nur Attribute in der Ergebnismenge, die die gegebenen Präferenzen erfüllen. Dabei kann für jede zusätzlich angegebene Präferenz die Zunahme des Wertes von M4 beobachtet werden. Für dieses Experiment wurde das Grundszenario allerdings insofern verändert, als die Dienste die Ergebnisse nicht mehr nach Jahr absteigend sortiert liefern, da sonst der Unterschied im Verhältnis zur Jahr-Präferenz nicht deutlich geworden wäre. Für die Sortierreihenfolge der Dienste wurde für dieses Experiment das Attribut Titel in absteigender Reihenfolge gewählt, da es in keiner offensichtlichen Beziehung zu den drei gegebenen Präferenzen steht. Als Ausführungsreihenfolge wurde die Strategie der Tiefensuche verwendet, die Präferenzen wurden jeweils mit dem Gewicht von 1 versehen.

13.6 Experiment 4: Zukünftige Entwicklungen

In diesem Experiment soll zukünftigen technischen und technologischen Entwicklungen Rechnung getragen werden. Es ist zu erwarten, dass schon bald sowohl die Antwortzeiten der Literaturdienste schneller werden (durch verringerte Latenzzeiten in den Netz-Komponenten) als auch eine größere parallele Anfragelast auf den Literaturdiensten zugelassen werden kann (durch leistungsfähigere Webserver-Architekturen und -Konzepte).

Im ersten Schritt soll nun der zulässige Parallelitätsgrad der Anfrageausführung erhöht werden. Dazu betrachten wir die Java-Anfrage des Grundszenarios. Beim vorliegenden Experiment wird die Anzahl der erlaubten parallelen Verbindungen zunächst bei allen Diensten in etwa verdoppelt. Anschließend wird nur der Parallelitätsgrad bei den Bibliotheken weiter erhöht und schließlich werden alle Dienste mit jeweils 100 parallelen Verbindungen befragt. Tabelle 13.12 zeigt die Ergebnisse dieses Experiments.

Schon der erste Schritt der Parallelitätserhöhung bringt eine deutliche Verbesserung im Antwortzeitverhalten (t_{max}). Dabei verbessert sich die Antwortzeit in etwa um den Faktor 2,5, d.h. im selben Verhältnis, in dem bei den Bibliotheksdiensten die Parallelität erhöht wurde. Das verdeutlicht nochmals die Tatsache, dass im gegebenen Grundszenario die Bibliotheksdienste sozusagen die langsamsten Dienste (vgl. z.B. auch Experiment 2, Abschnitt 13.4) und damit maßgeblich am Zustandekommen von t_{max} beteiligt sind.

Java-Anfrage	UBKA	BLB	AMAZON	LOB	t_{max} (Min:Sek od. Sekunden
# paralleler Verbindungen (Grundszenario)	10	10	25	25	2:42,9
# paralleler Verbindungen (1. Schritt)	25	25	50	50	1:06,9
# paralleler Verbindungen (2. Schritt)	50	50	50	50	37,7
# paralleler Verbindungen (3. Schritt)	100	100	100	100	35,4

Tabelle 13.12: Szenario mit erhöhtem Parallelitätsgrad der Literaturdienste (Java)

Im zweiten Schritt des Experiments wird lediglich der Parallelitätsgrad der Bibliotheksdienste weiter verdoppelt. In diesem Fall verringert sich wiederum die Antwortzeit t_{max}, sie kann allerdings nicht mehr im entsprechenden Maße verringert, nämlich halbiert werden. Der letzte Schritt des Experiments macht ebenfalls deutlich, dass keine beliebige Beschleunigung durch eine Erhöhung des Parallelitätsgrads möglich ist. Die Gründe dafür liegen zum einen in den gegebenen Abhängigkeiten der Informationsportionen, durch die es nicht möglich ist, alle Informationsportionen unabhängig voneinander parallel zu beschaffen. Zum anderen müssen die empfangenen Teilergebnisse in die Ergebnismenge integriert werden. Dieser Vorgang kann auch gerade bei großen Ergebnismengen nicht beliebig schnell durchgeführt werden.

Im nächsten Experiment sollen die Auswirkungen von beschleunigten Antwortzeiten seitens der zugrundeliegenden Dienste untersucht werden. Dazu wird wiederum die Java-Anfrage mit den Diensteigenschaften des Grundszenarios als Ausgangspunkt gewählt. Nun werden ebenfalls schrittweise die Antwortzeiten verringert, genauer gesagt die ersten beiden Parameter, die den zulässigen Bereich der Antwortzeiten beschreiben. Die anderen Parameter bleiben unverändert. Tabelle 13.13 zeigt die Ergebnisse dieses Experiments.

Java-Anfrage		UBKA	BLB	AMAZON	LOB	t_{max} (Min:Sek od. Sekunden)
Antwortzeitbereich (in Sekunden) (Grundszenario)	1. Ebene	0,8-1,9	2,5-5,0	3,0-10,0	1,0-3,0	2:42,9
	2./3. Ebene	1,5-3,5	2,5-4,5	0,5-3,0	0,8-2,0	
Antwortzeitbereich (in Sekunden) (1. Schritt)	1. Ebene	0,8-1,5	2,0-3,5	2,0-5,0	0,8-2,0	1:50,0
	2./3. Ebene	1,0-2,5	2,0-3,0	0,5-1,5	0,5-1,5	
Antwortzeitbereich (in Sekunden) (2. Schritt)	1. Ebene	0,5-1,0	0,5-1,0	0,75-3,0	0,5-1,5	55,9
	2./3. Ebene	0,5-1,0	0,5-1,0	0,5-1,0	0,5-1,0	

Tabelle 13.13: Szenario mit schnelleren Antwortzeiten der Literaturdienste (Java)

Im ersten Schritt des Experiments wurden die Antwortzeiten leicht nach unten korrigiert, so wie es eventuell in der nahen Zukunft erwartet werden kann. Dabei wurde bei den Antwortzeiten, bei denen der Minimalwert bereits recht gering war, lediglich der Maximalwert verringert, bei den anderen Antwortzeiten wurde sowohl der Minimal- als auch der Maximalwert erniedrigt. Dadurch konnte die Antwortzeit t_{max} allerdings um weniger als 1 Minute beschleunigt werden. D.h. auch in naher Zukunft ist es nicht zu erwarten, dass die Problematik der langsamen und den Benutzern nicht wirklich zumutbaren Antwortzeiten von Meta-Recherchesystemen verschwindet.

Im zweiten Schritt des Experiments wurden die Antwortzeiten nun in extremem Maße verringert, so dass nahezu alle Informationsportionen in weniger als einer Sekunde bezogen werden können. Lediglich bei AMAZON bzw. LOB konnten die entsprechenden Werte für die erste Informationsebene nicht derartig niedrig gewählt werden, da solche Werte angesichts von Informationsportionen mit 100 bzw. 50 Kurztiteln sehr unrealistisch erscheinen. Aber auch in diesem extrem beschleunigten Szenario, welches in der nahen Zukunft auch nicht wirklich zu erwarten ist, kann die Antwortzeit t_{max} die Marke von einer Minute nicht wesentlich unterschreiten.

Abschließend soll in diesem Experiment nun noch die Kombination von schnelleren Antwortzeiten und einem erhöhten zugelassenen Parallelitätsgrad betrachtet werden. Dazu wird die Java-Anfrage mit den Einstellungen aus Tabelle 13.12 und 13.13 durchgeführt, die jeweils im 1. Schritt und jeweils im 2. Schritt angegeben sind. Die Ergebnisse des Experiments sind in Tabelle 13.14 zu sehen.

Java-Anfrage	t_{max} (Sekunden)	t_{pref} (Sekunden)
Grundszenario	2:42,9	
paralleler und schneller (jeweils 1. Schritt)	46,1	
paralleler und schneller (jeweils 2. Schritt)	29,3	
paralleler und schneller (jeweils 1. Schritt) mit Jahr+Sprache-Präferenz		11,9

Tabelle 13.14: Zukünftige Szenarien ohne und mit Präferenzen (Java)

Durch die Kombination von schnelleren Antwortzeiten der Literaturdienste und einer erhöhten zugelassenen Parallelität bei der Anfrageausführung benötigt die Beantwortung der Java-Anfrage nun deutlich weniger Zeit als im gegebenen Grundszenario, welches charakteristisch für die heute existierende Situation ist. Durch das erste Szenario (jeweils 1. Schritt) wird sozusagen der technischen Entwicklung in der nahen Zukunft Rechnung getragen. D.h. es kann erwartet werden, dass eine derartige Anfrage von einem Meta-

Recherchesystem zwar wesentlich schneller beantwortet werden kann als heute, damit allerdings auch nicht unbedingt den Vorstellungen eines Benutzers hinsichtlich akzeptabler Antwortzeiten entspricht. Auch wenn wir in einem zweiten Szenario extrem schnelle Antwortzeiten und den höchsten Parallelitätsgrad, der noch zu Verbesserungen im Antwortzeitverhalten führt, annehmen (jeweils 2. Schritt), ändert sich die Situation nicht mehr wesentlich. D.h. die heute bestehende Problematik eines vollständig integrierten Meta-Recherchesystems wird sich auch durch den zu erwartenden technologischen Fortschritt nicht bewältigen lassen.

Zur Lösung dieser Problematik sind also zusätzliche Ansätze konzeptueller Art notwendig, wie beispielsweise der in der vorliegenden Arbeit vorgeschlagene Ansatz zur Unterstützung von Präferenzen. Werden beispielsweise im zukünftigen Szenario (jeweils 1. Schritt) zwei zusätzliche Präferenzen angegeben, so beträgt die Zeit, die der Benutzer auf die von ihm bevorzugten Dokumente warten muss, etwas mehr als 10 Sekunden. Dabei können die beiden gewählten Präferenzen auch heute bereits durch die Strategie der präferenzbasierten Anfragezerlegung unterstützt werden und damit die Lieferung der relevanten Ergebnisse in entscheidendem Maße beschleunigen (vgl. Experiment 3 in Abschnitt 13.5).

13.7 Ergebnisse und Handlungsempfehlungen

Nachdem nun eine Reihe von Experimenten durchgeführt worden sind, sollen hier noch einmal die Ergebnisse der Experimente zusammengefasst werden. Für weitere Details zu einem Großteil der Experimente sei zusätzlich noch auf [SO02b] verweisen. Anschließend werden aus den Ergebnissen der Experimente Empfehlungen für das Design von Meta-Recherchesystemen und Literaturdiensten abgeleitet.

13.7.1 Ergebnisse

Duplikaterkennung. Mit den Experimenten zur Duplikaterkennung kann zunächst die Annahme bestätigt werden, dass zahlreiche Dokumente gleichermaßen von unterschiedlichen Literaturdiensten vorgehalten werden, so dass aufgrund der vorhandenen Synergieeffekte der Nutzen und Mehrwert von Meta-Recherchesystemen durch die Experimente bestätigt werden kann. Etwa 2/3 der zu einer Anfrage gefundenen Dokumente sind Duplikate, so dass durch die Verschmelzung von Duplikaten zudem auch die Größe der Ergebnismenge deutlich (um mehr als 1/3) verringert werden kann.

Die Strategien der exakten Duplikaterkennung verursachen auch bei sehr großen Ergebnismengen kaum einen zusätzlichen Zeitaufwand. Werden zur Duplikaterkennung Attribute wie Titel oder Autor verwendet, ist die Qualität der Erkennungsleistung allerdings recht gering. Aufgrund von fehlerhaften Katalogeinträgen können damit in etwa nur weniger als die Hälfte der vorhandenen Duplikate erkannt werden.

Die Erkennungsleistung beim exakten Vergleich der ISBN-Attribute ist hingegen sehr gut – auch wenn damit unterschiedliche Auflagen eines Buches nicht als Duplikate betrachtet werden können. Allerdings können existierende Meta-Recherchesysteme diese Vorgehensweise nicht anwenden, da sie zu einer gegebenen Anfrage nur einen Teil der Informationen beschaffen und dieser Teil i.d.R. eben nicht die ISBN-Angabe der Dokumente enthält.

Die Erkennungsleistung der toleranten Vergleiche in Bezug auf das Titel- oder Autor-Attribut ist zwar deutlich besser als die der exakten Vergleiche, allerdings können auch mit toleranten Vergleichen nur etwa 2/3 der vorhandenen Duplikate identifiziert werden. Bei toleranten Vergleichen macht sich zudem bei mittel- bis sehr großen Ergebnismengen auch rasch der zusätzliche Zeitaufwand bemerkbar. Dabei verursacht die Implementierung der Damerau-Levenshtein-Metrik mit einer geringen Toleranzschwelle (DL80) bei sehr großen Ergebnismengen mit Abstand am wenigsten zusätzliche Verzögerungen. Bei Ergebnismengengrößen von etwa 500 Treffern kann mit dieser Strategie, aber auch mit der Trigramm-Methode (TRI95) eine tolerante Duplikaterkennung erfolgen, die keine signifikanten Auswirkungen auf das gesamte Antwortzeitverhalten der Anfragebearbeitung hat.

Anfrageausführung. Die Experimente zu den unterschiedlichen Strategien der Anfrageausführung zeigen, dass diese Strategien nicht im Allgemeinen bewertet werden können. Durch die vollständige Ausnutzung des erlaubten Parallelitätsgrads unterscheiden sie sich nicht im Hinblick auf die benötigte Gesamtzeit zur vollständigen Bearbeitung einer Anfrage. Sie unterscheiden sich lediglich in der Reihenfolge, in der die Informationen, d.h. die Attribute zu den Dokumenten geliefert werden. In den Experimenten konnten drei kritische Faktoren identifiziert werden, von denen die Eignung einer Ausführungsstrategie abhängt.

Ein kritischer Faktor ist die vertikale Informationsportionierung der Literaturdienste. Sind die Attribute in etwa gleichmäßig über die verschiedenen Informationsebenen verteilt und sind dem Benutzer keine Attribute besonders wichtig, so kann prinzipiell jede der drei Ausführungsstrategien (Breiten-, Tiefensuche, FCFS) gleichermaßen verwendet werden. Enthält allerdings eine Informationsebene wesentlich mehr Informationen als die anderen Informationsebenen, so ist der Strategie der Vorzug zu geben, die diese Informationsebene mit den meisten Attributen zu einem früheren Zeitpunkt besorgt. Ein zweiter kritischer Faktor kommt ins Spiel, sobald der Benutzer Vorlieben für gewisse Attribute angegeben hat. Dann ist diejenige Strategie vorzuziehen, die die Informationsebene, die die (meisten der) bevorzugten Attribute enthält, möglichst früh beschafft. Der dritte kritische Faktor ist die zu erwartende Antwortzeit für die unterschiedlichen Informationsebenen. In diesem Zusammenhang ist die Strategie am besten geeignet, die zuerst die Informationsebene beschafft, die schneller als die anderen verfügbar ist.

Da die genannten drei Faktoren nicht isoliert und unabhängig voneinander auftreten, muss jeweils für die Gesamtheit der Gegebenheiten die beste Strategie gewählt werden. Falls die Breiten- und die Tiefensuche in etwa gleich gut geeignet sind, kann die FCFS-Strategie

verwendet werden, da sie eine Mischform der beiden Strategien darstellt, die den Bedürfnissen des Benutzers besonders gut entgegen kommt.

Anfrageplanung. Durch die Experimente zur Strategie der präferenzbasierten Anfragezerlegung konnte gezeigt werden, dass sich mit jeder zusätzlich verfügbaren Präferenzangabe die Wartezeit des Benutzers, bis alle Informationen zu den bevorzugten Dokumenten eingetroffen sind, deutlich reduzieren lässt. Gerade bei Anfragen mit großen Ergebnismengen bringt die Formulierung von Präferenzen und der Einsatz der präferenzbasierten Anfragezerlegung den entscheidenden Vorteil, der die Antwortzeit in Bereiche lenkt, die als akzeptabel für den Benutzer gelten. Bei der sehr großen Java-Anfrage konnten beispielsweise durch die Angabe von drei zusätzlichen Präferenzen die bevorzugten Dokumente mit sämtlichen Informationen in weniger als 10 Sekunden beschafft werden.

Damit konnten die in Kapitel 4 formulierten Thesen der Arbeit nun auch quantitativ belegt werden. These 1 (S. 119) wurde durch die Konzeption und Umsetzung der generischen Suchmuster und des persönlichen, persistenten Arbeitsbereichs in Kapitel 7 belegt. Der quantitative Nutzen von zusätzlichen Angaben (seien es Bedingungen oder Präferenzen) konnte im Rahmen der Experimente (Tabelle 13.10 und 13.11) nachgewiesen werden. These 2 (S. 121) wurde durch die Konzeption und Umsetzung der präferenzbasierten Anfragezerlegung bewiesen und konnte ebenso durch die Ergebnisse der Experimente (Tabelle 13.11) quantitativ untermauert werden.

Ein Nachteil wurde bei der Strategie der präferenzbasierten Anfragezerlegung jedoch auch deutlich. Sie kann nur dann vollständig für alle gegebenen Präferenzen durchgeführt werden, wenn die Literaturdienste auch Anfragemöglichkeiten oder zumindest Sortieroptionen für die betreffenden Attribute anbieten. Gerade für die Lieferzeiten – die wie die Jahr- und Sprachangaben besonders gut für die Formulierung von Präferenzen geeignet sind – bieten Bibliotheksdienste zur Zeit noch keine derartige Unterstützung an.

Zukunftsprognosen. Die Experimente mit schnelleren Antwortzeiten und einem höheren zugelassenen Parallelitätsgrad seitens der Literaturdienste zeigen, dass sich die Problematik der vollständigen Anfragebearbeitung in Meta-Recherchesystemen auch in der nahen Zukunft nicht wesentlich verbessern wird. Zwar werden die benötigten Antwortzeiten dadurch geringer, aber sogar mit extrem schnellen Antwortzeiten und einer extrem hohen zugelassenen Parallelität, wie sie auch in ferner Zukunft noch nicht zu erwarten sind bzw. ist, können keine Antwortzeiten erreicht werden, die in einem als akzeptabel geltenden Bereich liegen.

Diese Ergebnisse bestätigen damit den Ansatz der vorliegenden Arbeit, die argumentiert, dass die Problematik der vollständigen Anfragebearbeitung nur durch die Entwicklung und den Einsatz von neuartigen Konzepten handhabbar gemacht werden kann. Der in dieser Arbeit vorgeschlagene Lösungsansatz zum Einsatz und zur Unterstützung von Präferenzen kann bereits in aktuellen Szenarien und natürlich auch in zukünftigen Szenarien die entscheidenden Verbesserungen im Antwortzeitverhalten herbeiführen.

13.7.2 Empfehlungen für das Design von Meta-Recherchesystemen

In diesem Abschnitt sollen nun die gewonnenen Erkenntnisse der Experimente in allgemeine Empfehlungen für das Design von Meta-Recherchesystemen umgesetzt werden.

- Duplikaterkennung: Jedes Meta-Recherchesystem sollte eine Möglichkeit vorsehen, die es erlaubt, die Duplikate unter den gefundenen Dokumenten zu identifizieren. Erst dadurch können die vorhandenen Synergien der einzelnen existierenden Literaturdienste für den Benutzer nutzbar gemacht werden. Zum einen können dadurch zu einem Dokument mehrere Informationen, d.h. inhaltliche Zusatzinformationen oder Beschaffungsalternativen, angezeigt werden, zum anderen kann dadurch die Ergebnismenge kleiner, kompakter und übersichtlicher gestaltet werden. Falls die ISBN-Information zu den Dokumenten vorhanden ist, wird auf jeden Fall diese Form der Duplikaterkennung empfohlen. Ansonsten sollten tolerante Duplikaterkennungsstrategien vorgesehen werden. Bewährt hat sich in der Praxis eine effiziente Implementierung der Damerau-Levenshtein-Metrik mit einer relativ niedrigen Toleranzschwelle. Bei Ergebnismengen von mehr als 500 oder gar 1000 Treffern sollte auf den Einsatz von toleranten Algorithmen verzichtet und ggf. auf exakte Operatoren zurückgegriffen werden. Es wird daher empfohlen, eine Reihe von unterschiedlichen Verfahren zur Duplikaterkennung bereitzuhalten und zu unterstützen, damit je nach Situation (d.h. Dokumentart, da Technische Berichte bekanntlich keine ISBN-Angabe haben, und Ergebnismengengröße) die geeignetste Form der Umsetzung ausgewählt werden kann.
- Vollständige Informationsbeschaffung: Die Beschaffung sämtlicher Informationen zu den gefundenen Dokumenten kann zum einen aufgrund der Ergebnisse von Abschnitt 5.3.2 empfohlen werden. Dort wurden die Merkmale von Dokumenten identifiziert, die die Benutzer für ihre Entscheidungen heranziehen. Gerade die Lieferbedingungen und im speziellen die Lieferzeit spielen für die Entscheidungen der Benutzers eine große Rolle. Diese Angabe ist allerdings bei Bibliotheksdiensten erst auf der dritten Informationsebene verfügbar. Zum anderen wird durch die vollständige Informationsbeschaffung erst der Einsatz der effektivsten und effizientesten Form der Duplikaterkennung, des exakten ISBN-Vergleichs, ermöglicht.
- Unterschiedliche Strategien der Anfrageausführung: Da es für jede der entwickelten Strategien zur Anfrageausführung (Breiten-, Tiefensuche und FCFS) Situationen gibt, in denen eine Strategie den anderen beiden Strategien überlegen ist, wird empfohlen, mehrere dieser Strategien (zumindest aber die Breiten- und die Tiefensuche) zu unterstützen, um diese dann bei Bedarf zur Anfrageausführung einsetzen zu können. Die Auswahl der besten Strategie kann beispielsweise dadurch geschehen, dass anhand der Metadaten von jedem Dienst im Vorhinein für jede Informationsebene berechnet wird, wie viele (ggf. auch mit Gewichten versehene) Attribute pro

Informationsportion im Verhältnis zur dafür benötigten durchschnittlichen Antwortzeit geliefert werden. Es wird dann diejenige Strategie zur Anfrageausführung ausgewählt, welche die Portionen der Informationsebene mit dem besten berechneten Verhältnis am frühesten beschafft. Da damit für unterschiedliche Literaturdienste ggf. unterschiedliche Ausführungsstrategien zu bevorzugen wären, sollte der in dieser Arbeit vorgeschlagene Ansatz zur Anfrageausführung zusätzlich dahingehend verfeinert werden, dass für jeden befragten Dienst eine entsprechende Ausführungsstrategie gewählt werden kann. Weiterhin kann es auch sinnvoll sein, dass noch weitere Mischformen der Breiten- und der Tiefensuche angeboten werden, beispielsweise eine Mischform, die für unterschiedliche Informationsebenen unterschiedliche Ausführungsreihenfolgen anwendet (z.B. die erste Ebene mit der Breitensuche und die zweite und dritte Ebene mit der Tiefensuche bearbeitet).

- Einsatz von Benutzerprofilen bzw. Präferenzen: Bei einem Meta-Recherchesystem sollte der Einsatz von Benutzerprofilen in dem Sinn unterstützt werden, dass der Benutzer persönliche Anfragen, Teile von Anfragen oder auch andere Angaben bzw. Einstellungen dauerhaft abspeichern und bei Bedarf im Zusammenhang mit neuen Anfragen wiederverwenden kann. Dadurch kann der Benutzer angeregt werden, die gestellten Anfragen mit mehr Angaben als nur einem aktuellen Stichwort zu formulieren. Diese zusätzlichen Angaben sind notwendig, um allzu große Ergebnismengen vermeiden und damit akzeptable Antwortzeiten liefern zu können. Sollten die Benutzer dennoch zu wenig Angaben (harte Bedingungen) spezifizieren, wird das Konzept der weichen Bedingungen (Präferenzen) empfohlen. Dadurch erhält die Anfragebearbeitung eine ausreichende Menge an Hinweisen, wie sie dem Benutzer die bevorzugten Dokumente möglichst schnell anzeigen kann, und der Benutzer bekommt schließlich immer sämtliche Dokumente zur gestellten Anfrage angezeigt, so dass er nicht befürchten muss, dass ihm dadurch relevante Dokumente entgehen. Ggf. ist es zu empfehlen, bei großen Ergebnismengen zumindest die Angabe von einigen Präferenzen vom Benutzer einzufordern. Der durchschlagende Erfolg, der schon beim Einsatz von wenigen (auch von einer oder von zwei) zusätzlichen Präferenzen zustande kommt, konnte durch die vorangegangenen Experimente bestätigt werden.

13.7.3 Empfehlungen für das Design von Literaturdiensten

Bei den Betrachtungen der Eigenschaften der Literaturdienste (Kapitel 8) sowie den gerade durchgeführten Experimenten wurden einige Beobachtungen gemacht, die im Folgenden in Form von Empfehlungen formuliert werden, aber lediglich als Hinweise oder Anregungen für zukünftige Entwicklungen im Bereich der Literaturdienste zu verstehen sind.

- Anfragemöglichkeiten: Existierende Literaturdienste stellen im Allgemeinen zwar eine Reihe von anfragbaren Attributen zur Verfügung (s. Tabelle 8.3), allerdings

werden nicht alle Attribute unterstützt, die Benutzer im Rahmen ihrer Entscheidungsfindung und damit bei der Formulierung von Präferenzen in Anspruch nehmen. Dadurch kann die erwiesenermaßen sehr wirksame Strategie der präferenzbasierten Anfragezerlegung gerade bei Präferenzen, die die Lieferzeit oder die Lieferkosten betreffen, nicht bei jedem Literaturdienst angewendet werden. Eine Erleichterung könnte auch bereits dadurch geschaffen werden, dass zumindest Ergebnissortierungen nach den entsprechenden Attributen angeboten werden.

- Informationsportionierung: Motivation für die Informationsportionierung der Literaturdienste ist sicherlich die größtenteils direkte und manuelle Nutzung der Dienste über ihre Webschnittstelle. Aber auch bei einer manuellen Recherche bereitet die Beschaffung der Ausleihzeiten von der dritten oder einer noch tieferen Informationsebene bei den meisten Bibliotheken große Mühen. Dabei sollte den Dienstanbietern bekannt sein, dass diese Angaben einen großen Einfluss auf die Entscheidungen der Benutzer haben. Daher wäre es sowohl für die direkten Benutzer eines Literaturdienstes als auch für die Integration eines Literaturdienstes in ein Meta-Recherchesystem wünschenswert, die verfügbaren Informationen maximal auf zwei Informationsebenen zu verteilen. Die Lieferzeit sollte zumindest auf der zweiten Informationsebene zu finden sein. Die ISBN-Angabe könnte hingegen auf der ersten Ebene zumindest bei der Nutzung durch ein Meta-Recherchesystem einen entscheidenden Fortschritt bringen. Ebenfalls wünschenswert wäre die Weiterentwicklung des Angebots von unterschiedlichen Anzeigeformaten, die mehr oder weniger Informationen auf den jeweiligen Informationsebenen präsentieren, wie es beispielsweise schon heute ansatzweise von ACM-DL oder CSBIB (s. Tabelle 8.3) angeboten wird. Dadurch könnte ein gleichzeitiges Nebeneinander von direkter Nutzung durch die Benutzer und indirekter Nutzung im Rahmen eines Meta-Recherchesystems unterstützt werden.

13.8 Resümee

In diesem Kapitel wurden eine Reihe von Experimenten zur Validierung der unterschiedlichen Anfragebearbeitungsstrategien durchgeführt. Dabei konnten die unterschiedlichen Strategien der Duplikaterkennung jeweils sowohl im Hinblick auf ihre Erkennungsleistung als auch bezüglich ihres zusätzlich verursachten Zeitbedarfs bewertet werden (Experiment 1). Anhand von unterschiedlichen Szenarien konnten geeignete Einsatzfelder für die unterschiedlichen Strategien zur Anfrageausführung ermittelt werden (Experiment 2). Durch verschiedene Arten bzw. Kombinationen von Präferenzen konnte der Nutzen der vorgeschlagenen Strategie der präferenzbasierten Anfragezerlegung bereits beim Einsatz von nur wenigen Präferenzen gezeigt werden (Experiment 3). Damit konnten die in Kapitel 4 formulierten Thesen der Arbeit nun auch quantitativ nachgewiesen und belegt

werden. Anschließend konnte auch angesichts der zu erwartenden technischen und technologischen Entwicklungen nachgewiesen werden, dass diese Verbesserungen nicht ausreichend sind, um die Problematik einer vollständigen Anfragebearbeitung bei Meta-Recherchesystemen bewältigen zu können. Mit dem in dieser Arbeit vorgeschlagenen Lösungsansatz lassen sich allerdings bereits schon in der heutigen Situation die entscheidenden Verbesserungen erzielen, die den Einsatz von vollständig integrierten Meta-Recherchesystemen in der Praxis ermöglichen (Experiment 4).

Anschließend wurden die Ergebnisse der Experimente nochmals zusammengefasst. Dabei konnten für jede der vorgeschlagenen Strategien zur Anfragebearbeitung geeignete Szenarien bzw. Situationen identifiziert und benannt werden, in welchen diese am gewinnbringendsten eingesetzt werden können und sollten. Aus den Ergebnissen der Experimente wurden schließlich allgemeine Handlungsempfehlungen für das Design von Meta-Recherchesystemen und Literaturdiensten abgeleitet.

Dieses Kapitel beschließt den dritten und letzten Teil der vorliegenden Arbeit. Das nächste und letzte Kapitel rundet die vorliegende Arbeit ab. Es fasst dazu noch einmal die Ausgangsproblematik, den vorgeschlagenen Lösungsansatz und die erzielten Ergebnisse zusammen und gibt einen Ausblick.

14 Zusammenfassung und Ausblick

14.1 Zusammenfassung

14.1.1 Ausgangssituation

Im Bereich der Digitalen Bibliotheken bzw. der Literaturrecherche stehen dem Benutzer mittlerweile zahlreiche einzelne Recherche- und Beschaffungsdienste zur Verfügung. Angesichts der vorhandenen Dienste- und Angebotsvielfalt ist es für den Benutzer jedoch schwierig, die Menge der existierenden Dienste und Dienstangebote zu überblicken und für seine Informationsbedürfnisse jeweils die besten verfügbaren Angebote zu finden. Meta-Recherchesysteme – bereits bekannt und erfolgreich eingesetzt im Bereich der Suchmaschinen für das WWW – stellen in einer derartigen Situation einen geeigneten Lösungsansatz dar. Durch die Integration der Einzeldienste bieten sie dem Benutzer einen einheitlichen Zugangspunkt für seine Informationssuche an, über den er sämtliche verfügbare Angebote ermitteln und miteinander vergleichen kann. Meta-Recherchesysteme ermöglichen es dem Benutzer, von den vielfältigen existierenden Angeboten auch entsprechend profitieren zu können.

Anhand eines typischen Szenarios wurden zunächst die Anforderungen an ein Meta-Recherchesystem für den Bereich der wissenschaftlichen Literaturrecherche und -beschaffung erarbeitet. Die identifizierten Anforderungen betrafen dabei im Wesentlichen zwei Themenschwerpunkte: Zum einen wurde eine geeignete Benutzerinteraktion gefordert, die auf der einen Seite hinreichend ausdrucksstark, auf der anderen Seite aber auch einfach und intuitiv zu bedienen sein soll. Zum anderen wurde eine geeignete Anfragebearbeitung gefordert, die die Ergebnisse vollständig beschaffen und integrieren, aber auch möglichst schnell liefern soll. Die Herausforderung bei der Erfüllung des Anforderungskatalogs bestand nun darin, einen Lösungsansatz zu finden bzw. zu entwickeln, der alle genannten Anforderungen gleichermaßen berücksichtigt, besonders weil die Anforderungen innerhalb jedes Themenschwerpunkts in einem gewissen Spannungsverhältnis zueinander standen.

Die Betrachtung existierender Meta-Recherchesysteme im Bereich der Literaturrecherche und -beschaffung machte entscheidende Defizite im Bereich der Vollständigkeit und der Integration der Ergebnisse deutlich. Die Literaturanalyse von bekannten I^3-Ansätzen zeigte, dass deren Verfahren zur Integration von heterogenen Informationsquellen aufgrund einer unterschiedlichen Integrationsproblematik größtenteils nicht gewinnbringend auf den Bereich der Literaturdienste übertragen werden können. Die Betrachtung von existierenden Personalisierungstechniken konnte wertvolle Hinweise zum Nutzen von und Umgang mit Benutzerprofilen im Hinblick auf eine geeignete Benutzerinteraktion geben, allerdings erwies sich

keines der Verfahren in Funktionalität und Wirkweise als angemessen für die Unterstützung der wissenschaftlichen Literaturrecherche. Die betrachteten graphischen Benutzungsschnittstellen konnten ebenfalls einige Hinweise für die Gestaltung einer geeigneten Benutzerinteraktion geben, auch wenn kein existierender Ansatz direkt übernommen werden konnte.

Der Handlungsbedarf für die vorliegende Arbeit bestand damit im Wesentlichen in der Bereitstellung von neuen technischen Lösungen für eine geeignete Anfragebearbeitung, die gleichermaßen vollständig und integriert, aber auch schnell arbeiten kann. Für die Gestaltung einer geeigneten Benutzerinteraktion sollte auf einige Konzepte existierender Lösungsansätze zurückgegriffen werden, allerdings bestand die Herausforderung dabei in der Entwicklung einer neuen und geeigneten Form der Personalisierung und in der geeigneten Anpassung der übernommenen Konzepte.

14.1.2 Lösungsansatz und Durchführung

Zunächst beschäftigte sich der vorgeschlagene Lösungsansatz mit der Gestaltung einer geeigneten Benutzerinteraktion. Dazu wurde in Anlehnung an SQL die ausdrucksmächtige Anfragesprache MLS-QL entwickelt, die es erlaubt, hinreichend präzise Anfragen an das Meta-Recherchesystem zu stellen. Die zentrale Idee des Lösungsansatzes bestand an dieser Stelle darin, neben den üblichen harten Bedingungen auch weiche Bedingungen, sogenannte Präferenzen, in der Anfragesprache zuzulassen und zu unterstützen. Ergebnisse von Benutzerstudien konnten zuvor belegen, dass Benutzer bei der Durchführung einer Literaturrecherche weniger harte Bedingungen, als vielmehr eine Reihe von Vorlieben berücksichtigen. Um die entwickelte ausdrucksmächtige Anfragesprache für den Benutzer gleichzeitig auch einfach und intuitiv handhabbar zu machen, wie es im Anforderungskatalog gefordert wird, wurde das Konzept der generischen Suchmuster entwickelt. Ein generisches Suchmuster besteht aus Teilen einer MLS-QL-Anfrage und enthält üblicherweise eine Reihe von Präferenzen oder anderen bevorzugten Anfrageeinstellungen. Jeder Benutzer kann sein eigenes Repertoire an generischen Suchmustern in individueller Weise zusammenstellen, pflegen und weiterentwickeln. Bei Bedarf kann der Benutzer dann eine aktuelle Rechercheanfrage mit einem vorhandenen generischen Suchmuster anreichern und somit präzisieren. Im Rahmen des Lösungsansatzes wurde eine graphische Benutzungsschnittstelle realisiert, die sowohl die entwickelte Anfragesprache als auch das Konzept der generischen Suchmuster in geeigneter Weise umsetzt und somit für den Benutzer einfach und intuitiv handhabbar macht. Für die Ergebnispräsentation wurde das bereits bekannte HiCites-Anzeigeformat übernommen, welches dem Benutzer ein schnelles und direktes Vergleichen der gefundenen Ergebnisse ermöglicht.

Der zweite Teil des Lösungsansatzes betraf die Konzeption und Umsetzung einer Anfragebearbeitung, die dem zweiten Schwerpunkt des Anforderungskatalogs gerecht werden kann. Dazu wurden zunächst eine Reihe von Untersuchungen und Experimenten bezüglich

der strukturellen Eigenschaften und des Antwortzeitverhaltens existierender Literaturdienste durchgeführt, um die gegebene Problematik der Informationsportionierung exakter präzisieren zu können. Daraufhin wurde eine geeignete Modularisierung der Anfragebearbeitung für ein Meta-Recherchesystem entworfen, bestehend aus den drei Hauptkomponenten der Anfrageplanung, der Anfrageausführung und der Ergebnisintegration. Bei jeder dieser Komponenten konnten gewisse Handlungsspielräume identifiziert werden, für welche prinzipiell unterschiedliche Verfahren oder Strategien entwickelt und eingesetzt werden können. Der Lösungsansatz bestand nun im weiteren Verlauf darin, für jede dieser Komponenten eine Reihe von geeigneten Verfahren oder Strategien zu entwickeln. Anschließend sollte untersucht werden, welche der entwickelten Verfahren sich als geeignet erweisen können.

Für die Anfrageplanung wurde die Strategie der präferenzbasierten Anfragezerlegung entwickelt, die es erlaubt, die vom Benutzer bevorzugten Dokumente sehr schnell zu beschaffen. Zur Anfrageausführung wurden unterschiedliche Vorgehensweisen, d.h. Reihefolgen festgelegt, in denen die einzelnen Informationsportionen der Literaturdienste beschafft werden. Unabhängig von der Reihenfolge werden dabei jeweils die von den Literaturdiensten vorgegebenen Lastbeschränkungen eingehalten und ausgeschöpft. Zur Duplikaterkennung können unterschiedliche Attribute, aber auch unterschiedliche Verfahren zum Vergleichen der Attributwerte (exakte und tolerante Vergleiche) verwendet werden. Dazu wurden drei unterschiedliche Algorithmen für tolerante Zeichenkettenvergleiche realisiert, zwei bereits bekannte Verfahren und ein neuer Algorithmus, der die Vorteile der beiden anderen Verfahren hinsichtlich der Erkennungsleistung vereint. Da sich die Eignung der unterschiedlichen entwickelten Strategien nur schwer auf konzeptioneller Ebene beurteilen lässt, wurden anschließend geeignete Bewertungsmaße definiert, anhand derer praktische Untersuchungen und Beurteilungen durchgeführt werden konnten. In den Bewertungsmaßen spiegeln sich die Ergebnisse zahlreicher Benutzerstudien wider, in denen herausgefunden wurde, dass Benutzer möglichst schnell möglichst viele Informationen erhalten möchten und dabei gewisse Angaben bevorzugen bzw. aufgrund von gewissen Angaben ihre Entscheidungen treffen.

Im dritten Teil des Lösungsansatzes wurde zunächst eine Simulationsumgebung entwickelt, die es ermöglicht, sowohl die strukturellen Eigenschaften als auch das Antwortzeitverhalten von Literaturdiensten nachzubilden. Durch die Simulation der befragten Literaturdienste ist es möglich, die unterschiedlichen entwickelten Strategien der Anfragebearbeitung nicht nur in aktuellen, sondern auch in beliebigen Szenarien zu testen, die beispielsweise Dienste mit schnelleren Antwortzeiten oder einer höheren erlaubten Anfragelast enthalten können als es momentan in der Realität der Fall ist.

Schließlich wurde eine Reihe von Experimenten durchgeführt, in denen zunächst für jede der drei Komponenten der Anfragebearbeitung nacheinander die unterschiedlichen Strategien untersucht und miteinander verglichen werden konnten. Dabei wurden für jede entwickelte Strategie Szenarien gefunden, in denen sie hinsichtlich der Bewertungsmaße deutlich besser

abschnitt als die anderen möglichen Strategien. Daher wurde anschließend ein Rahmenwerk für ein Meta-Recherchesystem vorgeschlagen, welches sämtliche der entwickelten Strategien enthält, zusammen mit einem Satz von Anwendungsempfehlungen für die einzelnen Strategien. Durch Experimente mit Szenarien, die mögliche Entwicklungen der nahen Zukunft widerspiegeln, konnte gezeigt werden, dass auch mit Literaturdiensten, die deutlich schneller antworten und wesentlich mehr Anfragelast erlauben, die Problematik der Informationsportionierung nicht gelöst werden kann. Hingegen konnte der in dieser Arbeit vorgeschlagene Lösungsansatz, genauer gesagt der Einsatz von Präferenzen, validiert werden. Durch den Einsatz von Präferenzen ist es möglich, die Anforderungen an die Anfragebearbeitung zu erfüllen, d.h. dem Benutzer sowohl vollständige und integrierte als auch schnell verfügbare Ergebnisse zu liefern. Durch die Experimente konnte gezeigt werden, dass schon wenige, d.h. zwei bis maximal drei zusätzliche Präferenzen ausreichen, um Antwortzeiten von mehreren Minuten in einen für den Benutzer akzeptablen Bereich (von etwa 10 Sekunden) zu bringen.

14.2 Ausblick

14.2.1 Weiterentwicklung

Weitere Experimente. Im Rahmen der vorliegenden Arbeit wurden bereits eine Reihe von aussagekräftigen Experimenten durchgeführt, die zum einen die Qualität und Leistung der entwickelten Anfragestrategien beurteilen als auch den vorgeschlagenen Lösungsansatz insgesamt validieren konnten. Bei den Experimenten wurden allerdings jeweils nur die unterschiedlichen Strategien einer Anfragekomponente untersucht und miteinander verglichen. Dadurch wurden nicht alle in SIMPSON möglichen Parametervariationen und Strategiekombinationen berücksichtigt und erprobt. Weitere Experimente könnten beispielsweise zusätzliche Parametervariationen oder das Zusammenspiel der Strategien von unterschiedlichen Komponenten betrachten, z.B. den Einfluss von unterschiedlichen Ergebnissortierungen der Literaturdienste oder von unterschiedlichen Ausführungsreihenfolgen auf die Strategien der Duplikaterkennung.

Verfeinerungen der Ausführungsstrategien. Im vorgeschlagenen Lösungsansatz wird die gewählte Ausführungsreihenfolge für alle befragten Dienste einer Anfrage einheitlich festgelegt und durchgeführt. Die Experimente haben allerdings verdeutlicht, dass es durchaus möglich sein kann, dass für jeden befragten Dienst eine unterschiedliche Ausführungsreihenfolge am besten geeignet ist. Daher sollte die Komponente der Anfrageausführung dahingehend erweitert werden, dass die gegebenen Ausführungsstrategien flexibler eingesetzt werden können. Diese Flexibilität kann beispielsweise auch die explizite Wahl einer Ausführungsreihenfolge für jede Informationsebene eines Dienstes betreffen.

Hilfswerkzeuge. Für den praktischen Einsatz des entwickelten Meta-Recherchesystems sollten noch zusätzliche klassische Hilfswerkzeuge für die Anfrageunterstützung in Recherchesystemen eingebunden werden, beispielsweise Wörterbücher oder (mehrsprachige) Thesauri. Da nicht alle zugrundeliegenden Literaturdienste eine Stoppwortliste oder eine Wortstammreduktion beinhalten bzw. anbieten, könnte die entsprechende Funktionalität dazu auch im Rahmen des Meta-Recherchesystems vorgesehen und unterstützt werden. Zusätzlich sollte auch zu allen Konzepten und Funktionen der Benutzungsschnittstelle eine detaillierte Online-Hilfe zur Verfügung gestellt werden.

Benutzerstudien. Die entwickelte Benutzungsschnittstelle sollte vor einem praktischen Einsatz zunächst noch durch einige Benutzerstudien und -beobachtungen hinsichtlich ihrer Intuitivität und Bedienbarkeit untersucht und ggf. angepasst werden.

14.2.2 Übertragbarkeit und Fortschritt

Anfragebearbeitung. Der vorgeschlagene Lösungsansatz zur Anfragebearbeitung lässt sich auf alle Arten von Meta-Recherchesystemen übertragen, die die zugrundeliegenden Dienste über ihre Webschnittstelle integrieren, insbesondere wenn bei diesen Diensten eine Informationsportionierung vorliegt. Das trifft allerdings bei den meisten elektronischen Produktkatalogen, Online-Auktionssystemen oder Reise- und Flugbuchungssystemen zu. In diesem Zusammenhang können auch die unterschiedlichen entwickelten Strategien zur Duplikaterkennung in flexibler Weise eingesetzt werden, da auch in diesen Szenarien die Semantik, welche Ergebnisse als Duplikate aufzufassen sind, nicht immer eindeutig bzw. in einheitlicher Weise festgelegt werden kann.

Einsatz von Präferenzen. Der Lösungsansatz zur Unterstützung von Präferenzen kann jeweils dann eingesetzt werden, wenn mit großen Ergebnismengen zu rechnen ist, im Allgemeinen aber auch keine Ergebnisse im Vorhinein ausgeschlossen werden können, da sie letztendlich vom Benutzer selbst bewertet werden müssen. Geeignete Szenarien hierfür bieten wiederum die gerade genannten elektronischen Produktkataloge, Online-Auktionssysteme, Reise- und Flugbuchungssysteme. Besonders bei Meta-Recherchesystemen ist mit großen Ergebnismengen zu rechnen. Das Konzept der Präferenzen kann aber ebenso bei Einzelsystemen mit einem größeren Datenbestand gewinnbringend eingesetzt werden. Die Objekte der Recherche sollten allerdings mit einer Reihe von Eigenschaften beschrieben werden, auf welche sich die Präferenzen dann beziehen können. Zur Unterstützung des Benutzers beim Umgang mit den Präferenzen kann zusätzlich das in der vorliegenden Arbeit entwickelte Konzept der generischen Suchmuster verwendet werden. Jeder Benutzer kann in einem persönlichen Bereich seine Vorlieben dauerhaft ablegen und jederzeit wiederverwenden. Die generischen Suchmuster und die darin enthaltenen Präferenzen stellen damit eine Realisierungsform der Nutzwert-basierten Personalisierung dar.

Generische Suchmuster. Im allgemeinen zielt der Beitrag des Konzepts der generischen Suchmuster darauf ab, die Benutzer hinsichtlich ihrer persönlichen Suchvorlieben und Suchstrategien zu sensibilisieren. Die Diskrepanz zwischen den kognitionswissenschaftlichen Erkenntnissen darüber, welche komplexen Prozesse und Entscheidungen bei einer Informationssuche innerhalb des Benutzers implizit ablaufen, steht zur Zeit noch in keinem Verhältnis zu dem in Benutzerstudien beobachteten expliziten Verhalten der Benutzer. Bei der ständig zunehmenden Menge an verfügbaren Informationen ist es allerdings in Zukunft unerlässlich, die Entwicklung der benötigten Fähigkeiten und Fertigkeiten des Benutzers für die Durchführung einer erfolgreichen Informationssuche anzuregen und zu unterstützen.

Literaturverzeichnis

[AK00] J. L. Ambite and C. A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformational approach. In: *Artificial Intelligence Journal,* 118(1/2), pp. 115-161, April 2000.

[Bal97] M. Balabanovic. An Adaptive Web Page Recommendation Service. In: *Proceedings of the 1st International Conference on Autonomous Agents (Agents-97)*, Marina del Ray, CA, pp. 378-385, 1997.

[BHC98] C. Basu, H. Hirsh, W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In: *Proceedings of the 15th National Conference on Artificial Intelligence and the 10th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-98)*, Madison, WI, pp. 714-720, 1998.

[BK02a] G. Barish and C. A. Knoblock. An Expressive and Efficient Language for Information Gathering on the Web. In: *Proceedings of the 6th International Conference on AI Planning and Scheduling (AIPS-02), Workshop: Is There Life Beyond Operator Sequencing? - Exploring Real-World Planning*, Toulouse, France, April 2002.

[BK02b] G. Barish and C. A. Knoblock. Speculative Execution for Information Gathering Plans. In: *Proceedings of the 6th International Conference on AI Planning and Scheduling (AIPS-02),* Toulouse, France, April 2002.

[BKLW99] S. Busse, R. Kutsche, U. Leser, H. Weber. Federated Information Systems: Concepts, Terminology and Architectures. Technische Universität Berlin, Forschungsberichte des Fachbereichs Informatik, Bericht Nr. 99-9, April 1999.

[BLG99] K. D. Bollacker, S. Lawrence, C. L. Giles: A System for Automatic Personalized Tracking of Scientific Literature on the Web. In: *Proceedings of the 4th ACM Conference on Digital Libraries (DL-99)*, Berkeley, CA, pp. 105-113, August 1999.

[BP00] D. Billsus and M. Pazzani. User Modeling and Adaptive News Access. In: *User-Modeling and User-Adapted Interaction*, 10(2), pp. 147-180, 2000.

[BP99] D. Billsus and M. Pazzani. A Personal News Agent that Talks, Learns and Explains. In: *Proceedings of the 3rd International Conference on Autonomous Agents (Agents-99)*, Seattle, WA, May 1999.

[BS00] I. N. Bronstejn and K. A. Semendjajew. Taschenbuch der Mathematik. 5. überarb. und erw. Auflage, Thun, Frankfurt am Main, 2000. ISBN 3-8171-2015-X

[BS97] M. Balabanovic and Y. Shoham. Fab: Content-Based, Collaborative Recommendation. In: *Communications of the ACM*, 40(3), pp. 66-72, 1997.

[BS98] W. Sander-Beuermann and M. Schomburg. Internet Information Retrieval: The Further Development of Meta-Searchengine Technology. In: *Proceedings of the 1998 Internet Summit of the Internet Society*, Genf, July 1998.

[Bur02] R. Burke. Hybrid Recommender Systems: Survey and Experiments. In: *User Modeling and User-Adapted Interaction*, 12(4), pp. 331-370, 2002.

[BW97] M. Wang Baldonado and T. Winograd. SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests. In: *Proceedings of the Conference on Human Factors in Computing Systems (CHI-97)*, Atlanta, GA, pp. 11-18, 1997.

[CCH92] J. P. Callan, W. Bruce Croft, S. M. Harding. The INQUERY Retrieval System. In: *Proceedings of the 3rd International Conference on Database and Expert Systems Applications (DEXA-92)*, Valencia, Spain, pp. 78-83, 1992.

[CGG00] B. Chidlovskii, N. Glance, A. Grasso. Collaborative Re-ranking of Search Results. In: *Proceedings of the AAAI Workshop on Artificial Intelligence for Web Search*, Austin, TX, 2000.

[Chr01a] M. Christoffel. Cooperations and Federations of Traders in an Information Market. In: *Proceedings of the AISB Symposium on Adaptive Agents and Multi-Agent Systems*, York, March 2001.

[Chr01b] M. Christoffel. Metadata-based Provider Selection in an Open Market Environment. In: *Proceedings of the 3rd International Workshop on Computer Science and Information Technologies*, Yangantau, Russia, September 2001.

[Chr99] M. Christoffel. A Trader for Services in a Scientific Literature Market. In: *Proceedings of the 2nd International Workshop on Engineering Federated Information Systems (EFIS-99)*, Kühlungsborn, May 1999.

[CM00] Z. Chen and X. Meng. Yarrow. A real-time client-side meta-search learner. In: *Proceedings of the AAAI Workshop on Artificial Intelligence for Web Search*, Austin, TX, 2000.

[CM01] Z. Chen and X. Meng. PAWS: Personalized adaptive web search. In: *Proceedings of the WebNet Conference*, Orlando, FL, October 2001.

[CNSP00] M. Christoffel, J. Nimis, B. Schmitt, S. Pulkowski, P. Lockemann. An Infrastructure for an Electronic Market of Scientific Literature. In: *Proceedings of the 4th IEEE International Baltic Workshop on Databases and Information Systems*, Vilnius, May 2000.

[Coh96] W. W. Cohen. Learning Trees and Rules with Set-valued Features. In: *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-96)*, Portland, OR, Vol. 1, pp. 709-716, 1996.

[Con97] S. Conrad. Föderierte Datenbanksysteme: Konzepte der Datenintegration. Springer, Berlin/Heidelberg, 1997. ISBN 3-540-63176-3

[CPSL99] M. Christoffel, S. Pulkowski, B. Schmitt, P. Lockemann, C. Schütte. The UniCats Approach – New Management for Books in the Information Market. In: *Proceedings of the International Conference IuK99 – Dynamic Documents*, Jena, 1999.

[CPWB97] S. B. Cousins, A. Paepcke, T. Winograd, E. A. Bier, K. Pier. The Digital Library Integrated Task Environment (DLITE). In: *Proceedings of the 2nd ACM Conference on Digital Libraries (DL-97)*, Philadelphia, PA, pp. 142-151, 1997.

[CS02] M. Christoffel and B. Schmitt. Accessing Digital Libraries as Easy as a Game. In: *Proceedings of the 2nd International Workshop on Visual Interfaces for Digital Libraries*, Portland, OR, LNCS 2539 (extended version), 2002.

[CSS02] M. Christoffel, B. Schmitt, J. Schneider. Semi-Automatic Wrapper Generation and Adaption: Living with Heterogeneity in a Market Environment. In: *Proceedings of the 4th International Conference on Enterprise Information Systems (EIS-02)*, Ciudad Real, Spain, April 2002.

[Dam64] F. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. In: *Communications of the ACM*, 7(3), pp. 171-176, 1964.

[Dat94] C. J. Date. A guide to the SQL standard: a user's guide to the standard relational language SQL. 3. edition, Addison-Wesley, Reading, MA, 1994. ISBN 0-201-55822-X

[DC99] Dublin Core Metadata Element Set, Version 1.1. Dublin Core Metadata Initiative Recommendation, Reference Description, 1999. http://dublincore.org/documents/1999/07/02/dces/

[Del89] T. Delsey. Authority Control in an International Context. In: B. B. Tillett (ed.). *Authority Control in the Online Environment: Considerations and Practices*. New York, Haworth Press, p. 25, 1989.

[DG97a] O. M. Duschka and M. R. Genesereth. Answering Recursive Queries Using Views. In: *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-97)*, Tucson, AZ, May 1997.

[DG97b] O. M. Duschka and M. R. Genesereth. Infomaster - An Information Integration Tool. In: *Proceedings of the International Workshop on Intelligent Information Integration (during the 21st German Annual Conference on Artificial Intelligence (KI-97))*, Freiburg, September 1997.

[DP98] R. Däßler and H. Palm. Virtuelle Informationsräume mit VRML. Informationen recherchieren und präsentieren in 3D. dpunkt Verlag, 1998. ISBN 3-920993-78-0

[Ebn02] M. Ebner. SQL lernen. 2. aktualisierte Auflage, Addison-Wesley, München, 2002. ISBN 3-8273-2025-9

[EH97] D. Ellis and M. Haugan. Modelling the information seeking patterns of engineers and research scientists in an industrial environment. In: *Journal of Documentation*, 53(4), pp. 384-403, September 1997.

[Eve99] B. Eversberg. Was sind und was sollen Bibliothekarische Datenformate. 3. Auflage, überarbeitete und erweiterte Neuausgabe, Universität Braunschweig, ersetzt und ergänzt die Buchausgabe von 1994, 1999. http://www.biblio.tu-bs.de/allegro/formate/

[FGK00] N. Fuhr, N. Gövert, C. Klas. An Agent-Based Architecture for Supporting High-Level Search Activities in Digital Libraries. In: *Proceedings of the 3rd International Conference of Asian Digital Libraries (ICADL-00)*, pp. 247-254, Taejon, Korea, 2000.

[FKSM02] N. Fuhr, C. Klas, A. Schaefer, P. Mutschke. Daffodil: An Integrated Desktop for Supporting High-Level Search Activities in Federated Digital Libraries. In: *Proceedings of the 6th European Conference on Digital Libraries (ECDL-02)*, Roma, Italy, LNCS 2458, September 2002.

[FLM98] D. Florescu, A. Y. Levy, A. O. Mendelzon. Database Techniques for the World-Wide Web: A Survey. In: *SIGMOD Record,* 27(3), pp. 59-74, 1998.

[FLM99] M. Friedman, A. Levy, T. Millstein. Navigational Plans for Data Integration. In: *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-99)*, Orlando, FL, pp. 67-73, July 1999.

[For99] P. J. Fortier. SQL-3: implementing the object relational database. McGraw-Hill, New York, 1999. ISBN 0-07-022062-X

[FOSB02] D. Fensel, B. Omelayenko, Y. Ding, E. Schulten, G. Botquin, M. Brown, A. Flett. Intelligent Information Integration in B2B Electronic Commerce. Kluwer Academic Publishers, 2002. ISBN 1-4020-7190-6

[FSG00] L. Fernández, J. A. Sánchez, A. García. MiBiblio. Personal Spaces in a Digital Library Universe. In: *Proceedings of the 5th ACM Conference on Digital Libraries (DL-00)*, San Antonio, TX, pp. 232-233, 2000.

[GBL98] C. L. Giles, K. D. Bollacker, S. Lawrence. CiteSeer: An Automatic Citation Indexing System. In: *Proceedings of the 3rd ACM Conference on Digital Libraries (DL-98)*, Pittsburgh, PA, pp. 89-98, 1998.

[HA00] J. M. Hellerstein and R. Avnur. Eddies: Continuously Adaptive Query Processing. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Dallas, TX, pp. 261-272, June 2000.

[Hem95] M. Hemmje. LyberWorld - A 3D Graphical User Interface for Fulltext Retrieval. In: *Proceedings of the Conference on Human Factors in Computing Systems (CHI-95)*, Denver, CO, Video summaries, May 1995.

[HKR00] J. Herlocker, J. Konstan, J. Riedl. Explaining Collaborative Filtering Recommendations. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW-00)*, Philadelphia, PA, pp. 241-250, December 2000.

[HKW94] M. Hemmje, C. Kunkel, A. Willet. LyberWorld - A Visualization User Interface Supporting Fulltext Retrieval. In: *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR-94)*, Dublin, July 1994.

[Höl00] C. Hölscher. Informationssuche im Internet: Web-Expertise und Wissenseinflüsse. Dissertation, Universität Freiburg, 2000.

[IFFL99] Z. G. Ives, D. Florescu, M. Friedman, A. Levy, D. S. Weld. An Adaptive Query Execution System for Data Integration. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, June 1999.

[ILWF00] Z. G. Ives, A. Y. Levy, D. S. Weld, D. Florescu, M. Friedman. Adaptive Query Processing for Internet Applications. In: *IEEE Data Engineering Bulletin*, 23(2), June 2000.

[JCM98] S. Jones, S. Cunningham, R. J. McNab. An Analysis of Usage of a Digital Library. In: *Proceedings of the 2nd European Conference on Digital Libraries (ECDL-98)*, Heraklion, Crete, LNCS 1513, pp. 261-277, 1998.

[JCMB00] S. Jones, S. Cunningham, R. J. McNab, S. J. Boddie. A transaction log analysis of a digital library. In: *International Journal on Digital Libraries*, 3(2), pp. 152-169, 2000.

[JFM97] T. Joachims, D. Freitag, T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In: *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97),* 1997.

[JP00] B. J. Jansen and U. Pooch. Web user studies: A review and framework for future work. In: *Journal of the American Society of Information Science and Technology*, 52(3), pp. 235-246, 2000.

[JSS00] B. Jansen, A. Spink, T. Saracevic. Real life, real users, and real needs: A study and analysis of user queries on the web. In: *Information Processing and Management*, 36(2), pp. 207-227, 2000.

[Kha02] I. I. Khalil (ed.). Proceedings of the 4th International Conference on Information Integration and Web-based Applications & Services (iiWAS-02). Bandung, Indonesia, SCS Publishing House, 2002.

[KHFH01] W. Kießling, B. Hafenrichter, S. Fischer, S. Holland. Preference XPath: A query language for e-commerce. In: *Information Age Economy*. 5. Internationale Tagung Wirtschaftsinformatik, Augsburg, Springer, pp. 427-440, September 2001.

[Kie02] W. Kießling. Foundations of Preferences in Database Systems. In: *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB-02)*, Hong Kong, August 2002.

[Kip97] N. A. Kipp. Case Study: Digital Libraries with a Spatial Metaphor. In: *Proceedings of SGML/XML Graphic Communications Association*, Washington, DC, pp. 631-640, 1997.

[KK01] W. Kießling and G. Köstler. Preference SQL – Design, Implementation, Experiences. Universität Augsburg, Report 7, 2001.

[Kli01] K. Kline. SQL in a Nutshell. 1. Auflage, O'Reilly, Köln, 2001. ISBN 3-89721-197-1

[KMAA01] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, I. Muslea, A. G. Philpot, S. Tejada. The ariadne approach to web-based information integration. In: *International Journal on Cooperative Information Systems (IJCIS) - Special Issue on Intelligent Information Agents: Theory and Applications*, 10(1/2), pp. 145-169, 2001.

[KMMH97] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. In: *Communications of the ACM,* 40(3), pp. 77-87, 1997.

[KST02] W. Kazakos, A. Schmidt, P. Tomczyk. Datenbanken und XML: Konzepte, Anwendungen, Systeme. Springer Xpert.press, Berlin/Heidelberg, 2002. ISBN 3-540-41956-X

[Kuh93] C. C. Kuhlthau. Seeking Meaning: a process approach to library and information services. Ablex Publishing Corp., Norwood, NJ, 1993.

[KW96] C. T. Kwok and D. S. Weld. Planning to Gather Information. In: *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96),* Portland, OR, August 1996.

[Lam95] L. Lamport: Das Latex-Handbuch. Addison-Wesley, Bonn, 1995. ISBN 3-89319-826-1

[LK97] E. Lambrecht and S. Kambhampati. Planning for Information Gathering: A Tutorial Survey. Arizona State University, Tech Report 96-017, May 1997.

[LKG99] E. Lambrecht, S. Kambhampati, S. Gnanaprakasam. Optimizing Recursive Information-Gathering Plans. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden, pp. 1204-1211, 1999.

[LMMP96] W. Lam, S. Mukhopadhyay, J. Mostafa, M. Palakal. Detection of Shifts in User Interests for Personalized Information Filtering. In: *Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR-96)*, Zurich, Switzerland, August 1996.

[LMSS95] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, D. Srivastava. Answering Queries Using Views. In: *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-95),* San Jose, CA, 1995.

[LRO96a] A. Y. Levy, A. Rajaraman, J. J. Ordille. Query answering algorithms for information agents. In: *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96),* Portland, OR, August 1996.

[LRO96b] A. Y. Levy, A. Rajaraman, J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In: *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB-96)*, Bombay, India, September 1996.

[Mar95] G. Marchionini. Information Seeking in Electronic Environments. Cambridge University Press, 1995. ISBN 0-521-44372-5

[MC00] M. Mahoui and S. Cunningham. A Comparative Transaction Log Analysis of Two Computing Collections. In: *Proceedings of the 4th European Conference on Digital Libraries (ECDL-00)*, Lisbon, Portugal, LNCS 1923, pp. 418-423, 2000.

[MC01] M. Mahoui and S. Cunningham. Search Behavior in a Research-Oriented Digital Library. In: *Proceedings of the 5th European Conference on Digital Libraries (ECDL-01)*, Darmstadt, LNCS 2163, pp. 13-24, 2001.

[MMPL96] S. Mukhopadhyay, J. Mostafa, M. Palakal, W. Lam, L. Xue, A. Hudli. An Adaptive Multilevel Information Filtering System. In: *Proceedings of the 5th International Conference on User Modeling (UM-96)*, Kona, HI, January 1996.

[Mön01] M. Mönnich. KVK – A Meta Catalog of Libraries. In: *European Research Libraries Cooperation*, Liber Quarterly, 11(2), pp. 121-127, 2001.

[MR00] R. J. Mooney and L. Roy. Content-Based Book Recommending Using Learning for Text Categorization. In: *Proceedings of the 5th ACM Conference on Digital Libraries (DL-00)*, San Antonio, TX, pp. 195-204, June 2000.

[MT89] S. Mukhopadhyay and M. A. L. Thathachar. Associative Learning of Boolean Functions. In: *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), pp. 1008-1015, 1989.

[NF00] F. Das Neves and E. A. Fox. A Study of User Behavior in an Immersive Virtual Environment for Digital Libraries. In: *Proceedings of the 5th ACM Conference on Digital Libraries (DL-00)*, San Antonio, TX, pp. 103-112, June 2000.

[Nie94] J. Nielsen. Usability Engineering. Morgan Kaufmann, San Francisco, 1994.

[Obe02] S. Oberländer. Simulation des Zeitverhaltens einer verteilten Anfrage an Digitale Bibliotheken. Diplomarbeit, Universität Karlsruhe, März 2002.

[Pat88] O. Patashnik. BibTeXing, February 1988. http://tex.loria.fr/bibdex/btxdoc.pdf

[PG99] A. Pretschner and S. Gauch. Ontology Based Personalized Search. In: *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-99)*, Chicago, IL, pp. 391-398, November 1999.

[PL00] R. Pottinger and A. Levy. A Scalable Algorithm for Answering Queries Using Views. In: *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB-00)*, Cairo, Egypt, 2000.

[PPF95] U. Pfeifer, T. Poersch, N. Fuhr. Searching Proper Names in Databases. In: R. Kuhlen and M. Rittberger (eds.). *Hypertext – Information Retrieval – Multimedia: Synergieeffekte elektronischer Informationssysteme* (*HIM-95)*, Konstanz, pp. 259-275, 1995.

[Pul01] S. Pulkowski. Technische Konzeption von Wandlern für den elektronischen Handel von Dokumenten. Dissertation, Universität Karlsruhe, Monsenstein und Vannerdat, Münster, February 2001.

[Qia96] X. Qian. Query folding. In: *Proceedings of the 12th International Conference on Data Engineering (ICDE-96)*, pp. 48-55, 1996.

[Rij79] C. J. van Rijsbergen. Information Retrieval. 2. Auflage, Butterworths, London, 1979. ISBN 0-408-70929-4

[RISB94] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In: *Proceedings of the ACM Conference on Computer Supported Collaborative Work (CSCW-94)*, Chapel Hill, NC, October 1994.

[Roc71] J. J. Rocchio. Relevance feedback in information retrieval in the SMART system. Prentice Hall, pp. 313-323, 1971.

[RR91] R. Rosenthal and R. Rosnow. Essentials of Behavioral Research: Methods and Data and Analysis. 2. edition, McGraw Hill, 1991.

[RS01] A. Rosenthal and L. Seligman. Scalability Issues in Data Integration, In: *Proceedings of the AFCEA Federal Database Conference*, San Diego, CA, 2001.

[Sal89] G. Salton. Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley, Reading, MA, 1989. ISBN 0-201-12227-8

[San98] W. Sander-Beuermann. Schatzsucher - Die Internet-Suchmaschinen der Zukunft. In: *c't*, 13/98, p. 178, 1998.

[SBC98] B. Shneiderman, D. Byrd, B. Croft. Sorting Out Searching. A User-Interface Framework for Text Searches. In: *Communications of the ACM*, 41(4), pp. 95-98, April 1998.

[Sch01] J. Schneider. Wandler in Digitalen Bibliotheken: semi-automatische Generierung und Evolutionsstrategien. Diplomarbeit, Universität Karlsruhe, September 2001.

[Sch02] B. Schmitt. Impact and Potential of User Profiles Used for Distributed Query Processing Based on Literature Services. In: *Proceedings of the PhD Workshop of the 8th International Conference on Extending Database Technology (EDBT-02)*, Praha, Czech Republic, LNCS 2490, March 2002.

[Sch98] A. Schmidt. Konzeption und Realisierung eines Informationsagenten für die Integration von Digitalen Bibliotheken im WWW. Studienarbeit, Universität Karlsruhe, November 1998.

[Sch99] A. Schmidt. Ein persönlicher Rechercheassistent für Digitale Bibliotheken im World-Wide Web. Diplomarbeit, Universität Karlsruhe, Dezember 1999.

[SHMM99] C. Silverstein, M. Henziger, H. Marais, M. Moricz. Analysis of a Very Large Web Search Engine Query Log. In: *SIGIR Forum*, 33(1), pp. 6-12, 1999.

[SM83] G. Salton and M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[SM95] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In: *Proceedings of the Conference on Human Factors in Computing Systems (CHI-95)*, Denver, CO, May 1995.

[SMFH01] M. A. Shah, S. R. Madden, M. J. Franklin, J. M. Hellerstein. Java Support for Data-Intensive Systems: Experiences Building the Telegraph Dataflow System. In: *ACM SIGMOD Record*, (30)4, December 2001.

[SO02a] B. Schmitt and S. Oberländer. Access Evaluation of Digital Libraries: Characteristics and Performance of Web OPACs. In: *Proceedings of* the *2nd International Workshop on New Developments in Digital Libraries (NDDL-02)*, Ciudad Real, Spain, 2002.

[SO02b] B. Schmitt and S. Oberländer. Evaluating and Enhancing Meta-Search Performance in Digital Libraries. In: *Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE-02)*, Singapore, December 2002.

[SPPC01] J. A. Sánchez, C. Proal, D. Pérez, A. Carballo. Personal and Group Spaces: Integrating Resources for Users of Digital Libraries. In: *Proceedings of the 4th Workshop on Human Factors in Computer Systems (IHC-01)*, Florianópolis, Brazil, pp. 183-194, 2001.

[SQL92] SQL2 Specification Draft. One of the last drafts of the SQL2 standard (July 1992) before it got accepted. http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

[SQL99] SQL3 ISO-ANSI Working Draft Database Language SQL/Foundation, August 1994. http://www.netaktive.com/biblio/sql/SQL98/sql-foundation-aug94.txt

[SR00] M. M. M. Snyman and M. J. van Rensburg. Revolutionizing name authority control. In: *Proceedings of the 5th ACM Conference on Digital Libraries (DL-00)*, San Antonio, TX, pp. 185-194, 2000.

[SS99] B. Schmitt and A. Schmidt. METALICA: An Enhanced Meta Search Engine for Literature Catalogs. In: *Proceedings of the 2nd International Conference of Asian Digital Libraries (ICADL-99)*, Taipei, Taiwan, pp. 142-160, 1999.

[Sul01] D. Sullivan. WebTop Search Rage Study. The Search Engine Report, Februar 2001. http://www.searchenginewatch.com/sereport/index.html

[TG74] J. T. Tou and R. C. Gonzales. Pattern Recognition Principles. Addison Wesley, 1974.

[Ull97] J. D. Ullman: Information Integration Using Logical Views. In: *Proceedings of the International Conference on Database Theory (ICDT-97)*, Delphi, Greece, LNCS 1186, pp. 19-40, January 1997.

[Wan97] P. Wang. The design of document retrieval systems for academic users: Implications of studies on users' relevance criteria. In: *Proceedings of the 60th ASIS Annual Meeting*, Washington, DC, pp. 162-173, 1997.

[Wie96] G. Wiederhold (ed.). Intelligent Integration of Information. Kluwer Academic Publishers, 1996. ISBN 0-7923-9726-6

[Win01] W. Winiwarter (ed.). Proceedings of the 3rd International Conference on Information Integration and Web-based Applications & Services (iiWAS-01), Linz, Austria, 2001. ISBN 3-85403-150-5

[XQU02] XQuery 1.0: An XML Query Language**,** last release 15 November 2002. http://www.w3.org/TR/xquery/

[YM99] T. Yan, H. Garcia-Molina. The SIFT Information Dissemination System. In: *ACM Transactions on Database Systems (TODS)*, 24(4), pp. 529-565, 1999.

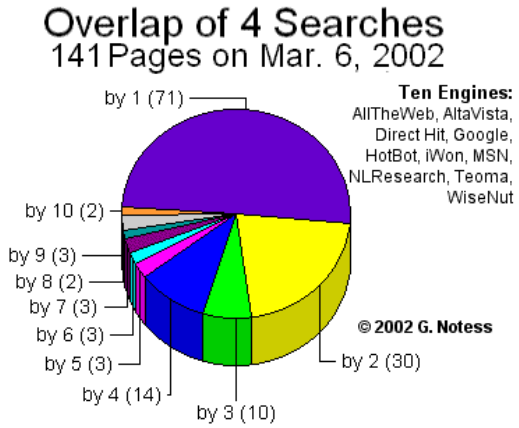
A Existierende Literaturdienste

Dienstkürzel	Dienstname	URL
ACM-DL	ACM Digital Library	http://www.acm.org/dl
AllBookStores	AllBookstores.com	http://www.allbookstores.com
AMAZON	Amazon, Deutschland	http://www.amazon.de
BLB	Badische Landes-bibliothek, Karlsruhe	http://sua.blb-karlsruhe.de
BUCH	buch.de	http://www.buch.de
CompuScience	Dienst vom FIZ Karlsruhe	http://stneasy.fiz-karlsruhe.de/html/deutsch/login1.html
CORA	Computer Science Research Paper Search Engine	http://www.cora.justresearch.com
CSBIB	Collection of Computer Science Bibliographies	http://liinwww.ira.uka.de/bibliography
Daffodil	Daffodil	http://www.daffodil.de
DBF	Deutsche Bibliothek, Frankfurt	http://dbf-opac.ddb.de
DBLP	Digital Bibliography & Library Project	http://dblp.uni-trier.de
GBV	Gemeinsamer Verbundkatalog	http://gso.gbv.de
IEEE-DL	IEEE Digital LIbrary	http://www.computer.org/publications
INSPEC	FIZ-Technik-Datenbank	(kostenpflichtig) http://www.fiz-technik.de/recherche/segment_elektrotechnik.htm
ITEC	FIZ-Technik-Datenbank	(kostenpflichtig) http://www.fiz-technik.de/recherche/segment_elektrotechnik.htm
KGK	Karlsruher Gesamtkatalog	http://www.ubka.uni-karlsruhe.de/hylib/ka_opac.html
KNO&KV	Koch, Neff & Ötinger GmbH und Köhler & Volckmar GmbH	http://www.buchkatalog.de
KVK	Karlsruher Virtueller Katalog	http://www.ubka.uni-karlsruhe.de/kvk.html

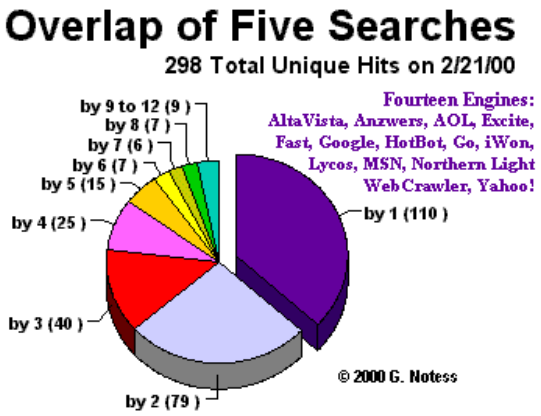
LEA	Lokales Elektronisches Aufsatzliefersystem	http://lea.ubka.uni-karlsruhe.de/lea
LOB	Lehmanns Online Bookshop	http://www.lob.de
LoC	Library of Congress, Washington, DC	http://lcweb.loc.gov
MeDoc	Multimediale Elektronische Dokumente	http://www.informatik.uni-stuttgart.de/medoc/medoc.html
NCSTRL	Networked Computer Science Technical Reference Library	http://www.ncstrl.org
NZDL	New Zealand Digital Library	http://www.nzdl.org/cgi-bin/library
RI	ResearchIndex	http://citeseer.nj.nec.com/cs
SCI	Science Citation Index	http://www.isinet.com/isi/products/citation/sci
Subito	Subito – Dokumente aus Bibliotheken	http://www.subito-doc.de
SWB	Südwestdeutscher Bibliotheksverbund	http://www.bsz-bw.de/wwwroot/opac.html
UBKA	Universitätsbibliothek, Karlsruhe	http://www.ubka.uni-karlsruhe.de

B Überlappungsgrad von Suchmaschinen

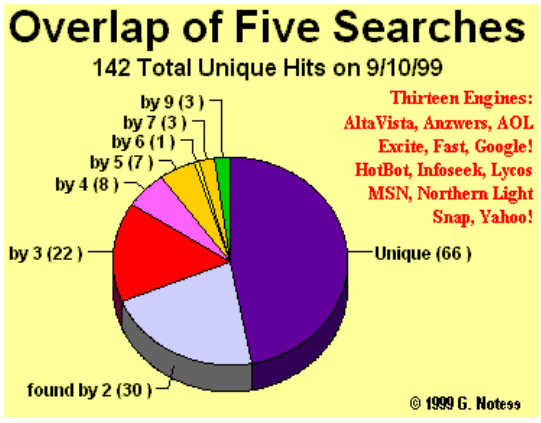
In diesem Abschnitt sind detailliertere Angaben der Showdownstudien aufgeführt. Die Grafiken stammen aus http://www.searchengineshowdown.com. Jedes Experiment wurde mit einer Reihe von Suchmaschinen durchgeführt (Engines), wobei alle Suchmaschinen jeweils mit vier oder fünf Anfragen (Searches) getestet wurden. Die Menge der insgesamt gefundenen Webseiten ist jeweils über der Kuchengrafik angegeben (Pages oder Total Unique Hits). Diese Treffermenge wird nun in den Kuchengrafiken so aufgetragen, dass man sehen kann, wie viele Webseiten jeweils nur von einer Suchmaschine gefunden wurde (by 1), wie viele Webseiten von genau zwei Suchmaschinen gefunden wurden (by 2), von drei (by 3), usw.



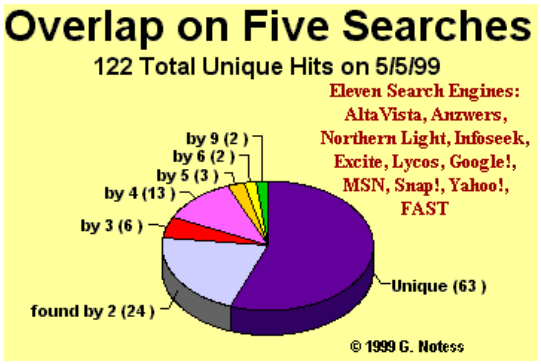
Showdownstudie März 2002



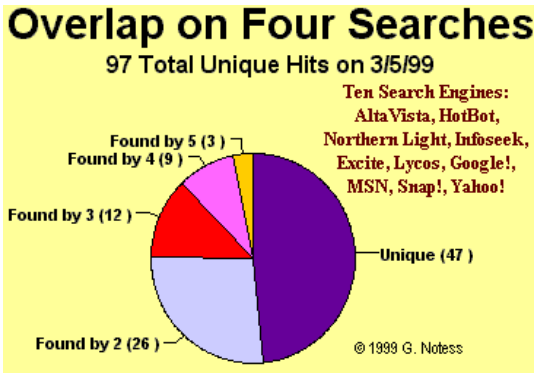
Showdownstudie Februar 2000



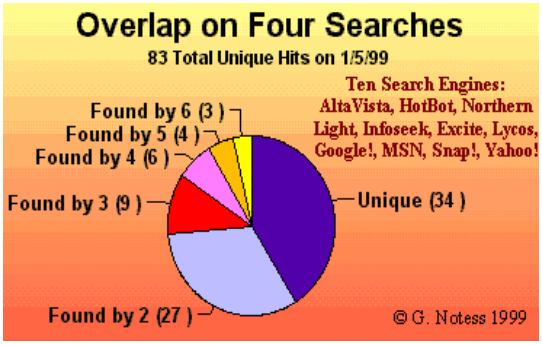
Showdownstudie September 1999



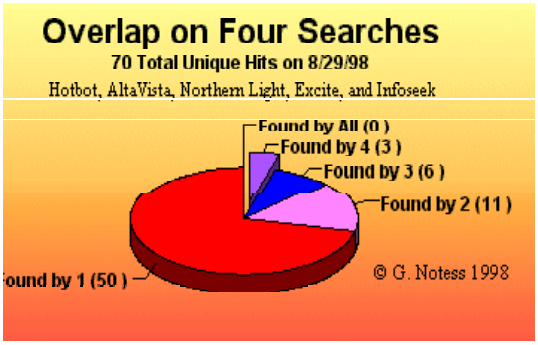
Showdownstudie Mai 1999



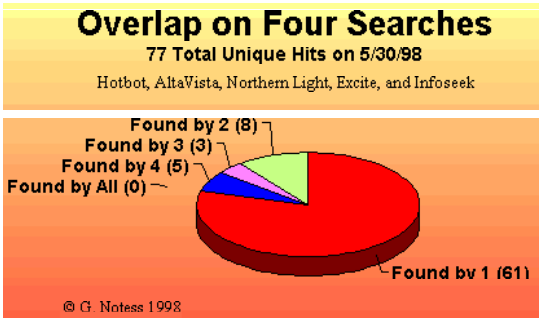
Showdownstudie März 1999



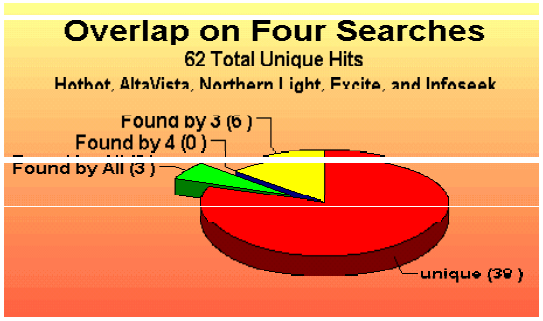
Showdownstudie Januar 1999



Showdownstudie August 1998



Showdownstudie Mai 1998



Showdownstudie Februar 1998

C Domänenmodell

C.1 DTD-Spezifikation des Domänenmodells

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Content: Domain Model for Integrated Literature Services -->
<!-- Contact: Bethina Schmitt -->

<!DOCTYPE Ergebnismenge [
<!ELEMENT Ergebnismenge ( Dokument+ )>

<!ELEMENT Dokument ( NachweisInfo+ , BibliographischeInfo? , ZusatzInfo? ,
BeschaffungsInfo* )>

<!ELEMENT NachweisInfo ( ID , Nachweisquelle )>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT Nachweisquelle (#PCDATA)>

<!ELEMENT BibliographischeInfo ( Titel? , Autoren? , Schlagworte? )>
 <!ELEMENT Titel ( Originaltitel? , Hauptsachtitel? , Titelzusatz? )>
  <!ELEMENT Originaltitel (#PCDATA)>
  <!ELEMENT Hauptsachtitel (#PCDATA)>
  <!ELEMENT Titelzusatz (#PCDATA)>
 <!ELEMENT Autoren ( Autor+ | Herausgeber+ | Koerperschaft )>
  <!ELEMENT Autor ( Vorname? , Nachname )>
  <!ELEMENT Herausgeber ( Vorname? , Nachname )>
   <!ELEMENT Vorname (#PCDATA)>
   <!ELEMENT Nachname (#PCDATA)>
  <!ELEMENT Koerperschaft (#PCDATA)>
 <!ELEMENT Schlagworte ( Schlagwort* , Systematik* )>
  <!ELEMENT Schlagwort (#PCDATA)>
  <!ELEMENT Systematik ( Systematikname , Kategorie+ )>
   <!ELEMENT Systematikname (#PCDATA)>
   <!ELEMENT Kategorie (#PCDATA)>
<!ATTLIST BibliographischeInfo Jahr CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Verlagsname CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Verlagsort CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Auflagenummer CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Auflageart CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Ausgabe CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Pubtyp (Buch|Artikel|...|DA) #IMPLIED>
<!ATTLIST BibliographischeInfo UebergeordnetesWerk CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Sprache CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Seiten CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Begleitmaterial CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo ISXN CDATA #IMPLIED>
<!ATTLIST BibliographischeInfo Preis CDATA #IMPLIED>
```

```
<!ELEMENT ZusatzInfo ( Beschreibungen? , Beurteilungen? , Verkaufsrang? ,
Zitiert? )>
 <!ELEMENT Beschreibungen ( Beschreibung* )>
 <!ATTLIST Beschreibungen Anzahl CDATA #IMPLIED>
  <!ELEMENT Beschreibung (#PCDATA)>
 <!ELEMENT Beurteilungen ( Beurteilung* )>
 <!ATTLIST Beurteilungen Anzahl CDATA #IMPLIED>
 <!ATTLIST Beurteilungen Durchschnittswert CDATA #IMPLIED>
  <!ELEMENT Beurteilung (#PCDATA)>
 <!ELEMENT Verkaufsrang ( Rang* )>
 <!ATTLIST Verkaufsrang Durchschnittswert CDATA #IMPLIED>
  <!ELEMENT Rang (#PCDATA)>
 <!ELEMENT Zitierung ( Von* , Nach* )>
 <!ATTLIST ZitiertVon Anzahl CDATA #IMPLIED>
 <!ATTLIST ZitiertNach Anzahl CDATA #IMPLIED>
  <!ELEMENT Von (#PCDATA)>
  <!ELEMENT Nach (#PCDATA)>
<!ATTLIST ZusatzInfo Titelbild CDATA #IMPLIED>
<!ATTLIST ZusatzInfo Inhaltsverzeichnis CDATA #IMPLIED>

<!ELEMENT BeschaffungsInfo ( Volltext* )>
 <!ELEMENT Volltext ( Format , Text )>
  <!ELEMENT Format (#PCDATA)>
  <!ELEMENT Text (#PCDATA)>
<!ATTLIST BeschaffungsInfo Lieferzeit CDATA #REQUIRED>
<!ATTLIST BeschaffungsInfo Lieferkosten CDATA #REQUIRED>
<!ATTLIST BeschaffungsInfo Lieferart (Ausleihe|Kauf|...|Online) #REQUIRED>
<!ATTLIST BeschaffungsInfo Lieferadresse CDATA #REQUIRED>
<!ATTLIST BeschaffungsInfo Lieferdienst CDATA #REQUIRED>
]>
```

C.2 XML-Repräsentation eines Dokuments

```
<Ergebnismenge>
<Dokument>

<NachweisInfo>
   <ID>15</ID>
   <Nachweisquelle>    AMAZON      </Nachweisquelle>
</NachweisInfo>
<NachweisInfo>
   <ID>7</ID>
   <Nachweisquelle>    UBKA        </Nachweisquelle>
</NachweisInfo>

<BibliographischeInfo Jahr="2001" Verlagsname="Teubner Verlag"
Auflagenummer="3" Ausgabe="gebunden" Pubtyp="Buch" Sprache="de"
Seiten="892" ISXN="3519226421" Preis="36,00">

<Titel>
   <Originaltitel> Java als erste Programmiersprache. - vom Einsteiger
                   zum Profi
   </Originaltitel>
   <Hauptsachtitel>    Java als erste Programmiersprache.
   </Hauptsachtitel>
```

```
    <Titelzusatz> - vom Einsteiger zum Profi
    </Titelzusatz>
</Titel>
<Autoren>
    <Autor>
        <Vorname>   Joachim    </Vorname>
        <Nachname>  Goll       </Nachname>
    </Autor>
    <Autor>
        <Vorname>   Cornelia   </Vorname>
        <Nachname>  Weiss      </Nachname>
    </Autor>
    <Autor>
        <Vorname>   Frank      </Vorname>
        <Nachname>  Mueller    </Nachname>
    </Autor>
</Autoren>
<Schlagworte>
    <Schlagwort> Computer & Internet </Schlagwort>
</Schlagworte>
</BibliographischeInfo>

<ZusatzInfo Titelbild="http://images-eu.amazon.com/images/P/3519226421.03
.LZZZZZZZ.jpg" Inhaltsverzeichnis="http://www.amazon.de/exec/obidos/tg/
stores/detail/-/books/3519226421/contents/ref=ed_tr_dp_2/302-6342760-
7604042">

<Beschreibungen Anzahl="0">
<Beurteilungen Anzahl="18" Durchschnittswert="4,5">
<Verkaufsrang>
    <Rang>  390    </Rang>
</Verkaufsrang>
</ZusatzInfo>

<BeschaffungsInfo Lieferzeit="0T:1S:0M" Lieferkosten="0,00"
Lieferart="Ausleihe" Lieferadresse="UBKA, EG" Lieferdienst="UBKA">
</BeschaffungsInfo>

<BeschaffungsInfo Lieferzeit="0T:0S:0M" Lieferkosten="0,00"
Lieferart="Einsehen" Lieferadresse="UBKA, 1. OG" Lieferdienst="UBKA">
</BeschaffungsInfo>

<BeschaffungsInfo Lieferzeit="3T:0S:0M" Lieferkosten="36,00"
Lieferart="Kauf" Lieferadresse="Büro" Lieferdienst="AMAZON">
</BeschaffungsInfo>

</Dokument>
</Ergebnismenge>
```