Diplomarbeit

Die Integration einer verlagsübergreifenden Suche in E-Books in ein lokales Bibliotheksportal

Entwicklung und Implementierung eines Prototyps zur Anbindung an Exlibris Metalib über das SRU-Protokoll

Vorgelegt zur Erlangung des akadem. Grades
Diplombibliothekar (FH)
an der Fachhochschule Potsdam,
Fachbereich Informationswissenschaften

Verfasser:
Florian Heß (Matr. 5810)
e-Mail: florianhess83@arcor.de
Vorgelegt am 01.08.2007

Gutachten:

1. Prof. Dr. Stephan Büttner, Fachhochschule Potsdam
2. Dr. Helmut Voigt, Humboldt-Universität zu Berlin, Zentralbibliothek Naturwissenschaften

.

Inhaltsverzeichnis

1 Einleitung

1.1 Zur vorliegenden Arbeit

Die Hochschulbibliothek der Humboldt-Universität zu Berlin hat eine Reihe von E-Book-Kollektionen bei unterschiedlichen Verlagen lizenziert. Während der Zugriff auf E-Books durch die Lizenz- und Nutzungsverträge zwischen der Bibliothek und der Verlage geregelt ist, wird ihre tatsächliche Nutzung erheblich dadurch erschwert, dass die technische Suchvorrichtung vollständig von dem Verlag gehalten wird. Dies bezieht sich nicht nur auf das Datenbankverwaltungssystem, sondern vor allem auf die Schnittstelle, mit der der Nutzer in der Recherche interagiert, welches die Suchergebnisse darstellt und darüber hinaus zusätzliche Funktionen anbietet, zum Beispiel eine Suchhistorie oder die nachträgliche Veränderung einer Suche.

Da das Interface eine unabhängige Eigenentwicklung des jeweiligen Verlages ist, herrscht große Heterogenität. In der Absicht, anbieterunabhängig Information zu finden, muss der Nutzer jedes einzelne Online-Angebot anwählen, sich mit den angebotenen Suchmöglichkeiten hinreichend vertraut machen, die Suchmaske entsprechend der Semantik seines Problems füllen und die Ergebnisse auswerten.

Diese Verfahrensweise ist nicht nur lästig, sondern steht auch dem Zweck einer Bibliothek entgegen, Menschen schnellen und einfachen Zugang zu öffentlichen Informationsressourcen zu ermöglichen.

Ein Teilproblem dessen ist, ob und inwieweit lizenzierte Verlagsangebote zum Bestand der Bibliothek „gehören“. In Anbetracht der zu begrüßenden Entwicklung von einer bestandsorientierten zu einer ressourcen- und auch bedarfsorientierten Bibliothekspolitik verschiebt sich dieses Problem jedoch in Richtung der Frage, ob und wie Bibliotheken und Verlage zusammenarbeiten können. Hier steht der bibliothekarische Auftrag mit dem zu wahrenden Interesse der Verlage an Autonomie in Konflikt. Während hierüber immer noch Uneinigkeit herrscht, sollte dem Nutzer die Unterscheidung zwar unbedingt kommuniziert werden, zum Beispiel durch Partnerlogos, ihn aber nicht im Zugang zu Information behindern oder diesen erschweren.

Die Erleichterung des Zugriffs auf die verlagsseitigen E-Book-Angebote ist der allgemeine Zweck des Projektes *Gather**YS**e*[1], das im Rahmen der vorliegenden Arbeit durchgeführt wird. Das Projekt steht im Zusammenhang mit der Arbeit von Ludwig [Lud07]. Sie behandelt ausführlich die allgemeine Problematik der elektronischen E-Books und analysiert eingehend zahlreiche online verfügbare Verlagsangebote für diese Medienart. Wiederum unter Bezug auf das GatherYSe-Projekt geht sie ferner

[1] Namensbildung aus *Gather **y**our **s**pecialists for **e**lectronic books.* Kraft möglicher Zusammenarbeit mit anderen Entwicklern ist eine Weiterentwicklung denkbar, die eine Änderung zu *for (any kind of) **e**lectronic media* rechtfertigen würde.

auf die Benutzerschnittstelle des Bibliotheksportalsystems Exlibris Metalib ein und gibt Empfehlungen zur Positionierung des fertigen Projektes in dessen Oberfläche.

Konkretes Ziel der vorliegenden Arbeit und des Projektes ist, die verlagsübergreifende Suche in E-Books, wie sie von [Lud07] konzipiert und organisiert wurde, praktisch zu entwickeln, zu implementieren und in das Metalib-System zu integrieren.

Dazu werden ergebnisrelevante Probleme[2], deren Lösung und/oder Folgen erläutert. Außerdem werden Ratschläge und Richtlinien für die Entwicklung von Anbindungen weiterer Verlagsangebote gegeben. Die Architektur und die Abläufe innerhalb GatherYSe, sowie Aspekte der Kommunikation zwischen diesem und dem Portal werden eingehend erläutert. Die Anbindung jedes einzelnen E-Book-Anbieters wird nicht nur implementiert, sondern auch an den jeweiligen Stellen der Arbeit codezeilen-bezogen kommentiert.

Vor der eigentlichen Implementierung sollen die technischen Gegegebenheiten der verlagseigenen Suchdiensten analysiert werden, vor allem in Hinblick auf ihre Automatisierbarkeit. Besondere Berücksichtigung finden die Nutzungsbedingungen der Suchdienste. In der vorliegenden Arbeit werden diese Erkenntnisse jedoch zusammen mit der Implementierung behandelt.

Die Software wird in der Programmiersprache Perl[3] realisiert. Die Hardware ist ein älteres System[4], dass dem Autor in der Institution für die Arbeit im Haus zur Verfügung gestellt wurde. Für die tatsächliche Anwendung des fertigen Systems durch die Nutzer der Bibliothek sollte GatherYSe unbedingt auf leistungsfähigerer Hardware installiert werden.

Die Sprache im Quellcode ist durchgehend Englisch, um seine Wiederverwendung im internationalen Raum zu vereinfachen. Dies bezieht sich sowohl auf Bezeichner aller Art, als auch auf Kommentare und Dokumentation.

Bei der Implementierung soll dabei weitestmöglich[5] den Richtlinien von [Con06] gefolgt werden.

Es handelt sich um eine Projektarbeit, in der drei Viertel des Zeitrahmens der eigentlichen Entwicklung und Implementierung gewidmet werden müssen, und die Literaturrecherche nur geringe Priorität erfahren kann. Entsprechend ergibt sich eine kurze Bibliografie.

1.2 Konventionen

Schreibmaschinenschrift Text, der in einem direkten Zusammenhang mit der Systembedienung oder -programmierung steht, erscheint in `Schreibmaschinenschrift`.

[2] d. h. exklusiv jener, die auf selbstverschuldete und korrigierte Fehler im Entwicklungs- und Implementierungsprozess zurückgehen, z.B. das Vergessen der zweiten Hälfte eines regulären Ersetzungsausdrucks (s/Muster/Ersetzungstext/), das wegen irreführender Meldungen eine langwierige Fehlersuche nach sich ziehen kann.

[3] Version 5.8.8 built for i586-linux-thread-multi

[4] AMD K6-400, 384MB RAM, OpenSuSE 10.1 ohne KDE

[5] Einschränkungen bestehen vor allem da, wo dem Programmierer CPAN-Module nahegelegt werden, die trotz begrenzter Effektivität bei Nutzung eine weitere Systemvoraussetzung bilden würden

Kursivdruck Besondere Erstbenennungen und Hervorhebungen sind sind *kursiv* gedruckt.

Pfad/zu/Datei.dat relative Pfadangaben beziehen sich auf Dateien im GatherYSe-Verzeichnis.

$ Das Dollarzeichen signalisiert, dass ein Befehl in die Unix-Shell eingegeben werden muss. Gegebenenfalls steht der Pfad eines Verzeichnisses davor, in das vorher gewechselt werden muss.

(Z...) Dies bezieht sich auf eine Stelle in einem Perlmodul von GatherYSe. Möglich sind hierbei die Angabe einer einzelnen Zeile (Z123), eines Blocks mit mehreren Zeilen (Z123f./ff.) und eines Fragments (Z123-149).

1.3 Die beiliegende CD

Die beiliegende CD enthält folgende Dateien und Verzeichnisse:

1. `diplomarb.pdf` ist die vorliegende Arbeit in elektronischer Form.
2. Die Datei `GatherYSe-0.1.tar.gz` enthält die GatherYSe-Distribution.
3. Im Verzeichnis `configs-hub/` liegen die Konfigurationsdateien für die von der Humboldt-Universität lizenzierten E-Book-Kollektionen.
4. Das Verzeichnis `GysHTML` enthält die Quelltexte in wohlformatierter Form. Die Datei `index.html` enthält die Übersicht.

1.4 Danksagungen

Zuvorderst seien Herr Lüttgau und Tim Goverdovski (EDV-Abt.) für ihre Metalib-bezogene Unterstützung gedankt. Ohne ihre Arbeit – die, aus glücklichen Umständen, zu Projektbeginn bereits einer anderen Metalib-Ressource galt – könnte das Bibliothekssystem der Humboldt-Universität auf keine SRU-Quellen zugreifen, und das Projekt hätte nicht vollendet, geschweige ausgiebig aus der Nutzersicht getestet werden können. Auch zusammen mit Herrn Voigt ergaben sich fruchtbringende und motivierende Gespräche, die zudem halfen, den Überblick zu behalten.

Auch Wolfram Schneider vom Fachinformationszentrum Chemie sei gedankt für seine prompten Informationen per E-Mail.

Nicht zuletzt lieben Dank an meine Familie für den mentalen Rückhalt und die häufigen Erinnerungen an das *Jetzt* und an den Boden der Tatsachen.

2 Grundkonzeption

2.1 Erfahrungen aus Pittsburgh

Für die technische Umsetzung von GatherYSe relevante Erfahrungen wurden unlängst von der University of Pittsburgh's Health Sciences Library System (HSLS) gemacht [FBMP07]. Die Problemstellung war, die bibliografischen Nachweise von E-Books im OPAC um eine Volltextsuche zu ergänzen.

Die Notwendigkeit, relevante Online-Titel zu finden, um dann die Titel einzeln nach Informationen zu durchsuchen, bremst laut Foust das Retrieval und schränkt die Nutzungspotenziale der E-Books ein.

Das verwendete „Electronic Book Search" ist eine auf der Vivísimo Velocity, eine Entwicklungsumgebung für Retrieval-Applikationen, basierende Lösung. Die Auswahl geschah nicht auf eine Evaluation mehrerer Alternativen hin, sondern es waren bereits Lizenzen dieses Produktes erworben worden.

Foust führt aus, dass Verbundsuchmaschinen maximal nur soviel leisten können wie die einzelnen Datenbanken und Quellen. Die boolesche Suche etwa sei nicht immer selbstverständlich: 90% ihrer lizenzierten Anbieter verwendeten die konventionelle Google-like Syntax, nur 10% gebrauchten boolesche Operatoren. Die Datenumwandlung zwischen Quelle und Suche gestalte sich oft komplex und sei aufgrund von Änderungen des Ausgabeformats schwierig zu pflegen. Eine Herausforderung seien außerdem Authentifizierungsprobleme.

> While the literature does not contain a discussion of federated searching used with full-text e-books specifically, the more general literature confirmed the library's decision to apply federated search technology to this situation and offered insight as to where future problems might occur.
> [FBMP07, S. 1]

Die Literatur gab Foust et al. wenig Rat zur konkreten Realisierung einer Volltextsuche in E-Books, sondern beschränkte sich auf allgemeine Betrachtungen und Vermutungen.

Die Velocity-Software besteht aus drei Teilen:

1. Enterprise Search Engine: Indexierung und Retrieval von Volltexten aus verschiedenen Dokumentenkollektionen.

2. Content Integrator: Anfrage bei Datenquellen und Vereinigung der Treffer in einer Treffermenge

3. Clustering Engine: Dynamische Gruppierung der Informationen.

Von diesen Komponenten fand jedoch nur der *Content Integrator* Anwendung. Zuerst musste für jedes Paket ein eigenes Profil erstellt werden. Hierbei galt es, in einem „Sourcefile“ die Konfiguration auf die jeweiligen Charakteristika (Suche, Ergebnisse, Authentifizierung, Datenübetragung) abzustimmen.

Nach der Konfiguration des *Content Integrators* erfolgte die Testphase, um Fehler und unerwartete Ergebnisse aufzuspüren, und wurde begleitet von Feineinstellungsmaßnahmen.

Als ein schwieriges Problem erwies sich allgemein die in der Art und im Format der ausgegebenen Informationen heterogene Trefferanzeige.

Neben der eigentlichen Datenextraktion galt es außerdem, für das Clustering die Trefferanzeige auf eine vernünftige Menge zu beschränken. Hier musste zwischen der „Clusterability“ und der für das Retrieval benötigten Zeit abgewogen werden.

Ein Clustering gehörte nicht zu den Anforderungen an GatherYSe.

Es waren schließlich Vereinbarungen mit Anbietern notwendig, um für Foust et al. hinderliche Schranken aus dem Weg zu räumen. Ein Beispiel war hier ein Lizenzfenster, dass umgangen werden musste:

> In one case, a provider added an individual license agreement screen to which a user must respond before accessing the provider's resources; because this interfered with Electronic Book Search, HSLS contacted the provider for permission to bypass the page, which was granted.[FBMP07]

Obwohl viel schneller als alle Ressourcen einzeln manuell zu durchsuchen, hinken Verbundsuchen hinsichtlich der Geschwindigkeit den indexbasierten Suchen hinterher. Daher war es notwendig, zumindest eine Rückmeldung zur laufenden Suche anzuzeigen.

2.2 Vorbetrachtungen

Die automatische parallele Abfrage (Metasuche) geschieht über standardisierte Schnittstellen und Formate, zu welchen das traditionelle Z39.50-Protokoll zum Austausch von bibliothekarischen Metadaten gehört. Die von [Lud07] evaluierten Verlagskollektionen bieten neben dem Web-Interface von sich aus keine solchen standardisierten Protokolle an und machen es daher erforderlich, die Kommunikation auf einer niedrigeren, veränderlicheren und damit weniger nachhaltigen Ebene stattfinden zu lassen: Hierbei geht es zum Einen darum, eine Anfrage in der vom Inhalteanbieter erwarteten Form zu generieren und abzuschicken, und zum Anderen die Ergebnisdaten aus dem HTML-Code der Antwort zu extrahieren. Dieses Verfahren zur Extraktion von Inhalten aus Webseiten wird gemeinhin als „Webscraping“ bezeichnet. Vor allem jedoch werbefinanzierte kommerzielle Inhalteanbieter treten solchen Bestrebungen ablehnend gegenüber, da die Werbung bei der Extraktion nicht berücksichtigt wird und somit die Nutzer der Inhalte nicht erreicht. Daher kann und soll die verlagsübergreifende Suche in E-Books

einen bestimmten Informationsanbieter nicht abfragen, wenn dessen Nutzungs- oder Lizenzbestimmungen diese Art der Datenverwendung ausdrücklich verbieten und der Humboldt-Universität keine Sondererlaubnis erteilt wird.

Eine essentielle Maßgabe für das Projekt ist es, insbesondere den Volltextzugriff auf die E-Books über das Metalib-Portal zur Verfügung zu stellen. Produzenten, die Volltextsuche für ihre E-Book-Kollektion anbieten sind Springer, Oxford (4 Pakete) und NetLibrary, welche aufgrund dessen vorrangig berücksichtigt werden. Das Besondere bei Springer ist, dass die Humboldt-Universität im Namen des Friedrich-Althoff-Konsortiums mit diesem Verlag und dessen Hosting-Partner Metapress erfolgreich über eine regelmäßige Indexierung und der Bereitstellung der Retrievaltechnik durch das Fachinformationszentrum für Chemie (FIZ Chemie) verhandelt hat. Die Einbindung der Springer-E-Books kann so über eine nachhaltigere und besser den bibliothekarischen Bedürfnissen anpassbare XML-basierten Schnittstelle erfolgen. [Lud07] führt die Problematik dieser Drei-Partner-Kooperation detailliert aus.

Der Volltextzugriff stellt einen Trumpf dar gegenüber der Methode des einfachen sog. *Catalogue Enrichments*, bei welchem die Verlage Metadaten über ihre elektronischen Medien an Bibliotheken schicken, so dass diese damit ihren OPAC[1] um die entsprechenden Medieneinheiten erweitern und einfach auffindbar machen können. Die alleinige Suche in Metadaten schöpft jedoch nicht die Möglichkeiten elektronisch vorliegender Werke aus. Erst die Volltextsuche erlaubt beispielsweise, nach bestimmten Phrasen und Formulierungen zu recherchieren, und dies optional verbunden mit bibliografischen Kriterien wie dem Autor.

Ein abhängig vom jeweiligen Inhalteanbieter größeres oder kleineres Problem ist die Authentifizierung. Hierbei gibt es zwei wesentliche Arten: Zum einen die IP-Adressenbasierte Methode, bei welcher der statische IP-Adressbereich der lizenznehmenden Institution beim Anbieter registriert wird und der Zugriff auf die Inhalte automatisch für alle PC-Arbeitsplätze aus diesem IP-Adressbereich freigeschaltet ist. Zum Anderen gibt es die Login-Methode, bei der der einzelne Benutzer in eine Eingangsmaske eine spezielle User-ID[2] und das mit ihr assoziierte Passwort eingeben muss. Der Server schickt eine Session-ID[3] an den Nutzer zurück, bzw. werden mit ihr sämtliche Links versehen, die in den während der Sitzung aufgerufenen Seiten enthalten sind. Bei dieser Variante kann zu lange Inaktivität seitens des Benutzers dazu führen, dass die Session-ID auf dem Server verfällt und ein erneuter Login erforderlich ist.

Ein dritter Ansatz, das sogenannte *Single Sign-On*, versucht die Vorteile der beiden Ansätze zu vereinen. Beim Single Sign-On bzw. bei dessen konkreten Ausprägungen wie Schibboleth, Athens [Edu] u.a. ist nur ein Login bei einer zentralen Authentifizierungsstelle bzw. bei einem teilnehmenden Anbieter erforderlich. Alle Anbieter, die den entsprechenden Dienst unterstützen, schalten daraufhin den Zugriff auf die Inhalte frei, deren Nutzungsrechte der Anwender erworben hat. Die priorisierten Kollektionen verwenden die IP-basierte Authentifizierung, Oxford und NetLibrary zusätzlich

[1] Online Public Access Catalogue
[2] Name, Pseudonym, e-Mail-Adresse, oder Zahl
[3] Sitzungsidentifikationsnummer

die Authentifizierung über Athens.

Weil die verlagsübergreifende Suche in E-Books nicht als Metalib-Modul im engeren Sinne implementiert wird, sondern auf einem eigenen Server läuft, muss dieser Server entweder eine IP-Adresse aus dem Bereich der Institution besitzen, oder seine Adresse muss in einem Antrag oder eigenen Lizenzvertrag dem Provider zur Freischaltung mitgeteilt werden. Der einzige Client des Servers ist das Metalib-Portal der Humboldt-Universität. Darum liegt es in der Verantwortung des Portalsystems, dass nur authentifizierte Anfragen an den Server geschickt werden. Diese Einschränkung ist aber nur dann notwendig, wenn seitens des E-Book-Anbieters keine nochmalige Prüfung der IP-Adresse beim eigentlichen Abruf des Volltextes vorgenommen wird.

2.3 Konzeption des GatherYSe-Servers

Die technische Dokumentation der Metalib-APIs[4] wurde von Lüttgau hinsichtlich ihrer Qualität als nicht ausreichend beurteilt. Die interne Softwarestruktur von Metalib ist mit der Zeit sehr komplex geworden und mittlerweile gar auf eigens angepasste CPAN-Erweiterungen[5] angewiesen. Dadurch kann die Integration weiterer CPAN-Module, die für das Projekt verwendet, aber nicht entsprechend an die Besonderheiten des umgebenen Metalib-Systems angepasst werden, unter Umständen scheitern, indem die Module nicht funktionieren oder sich undefiniert verhalten.

GatherYSe wird demzufolge nicht als Modul zur direkten Integration in das Metalib-System implementiert, sondern als ein eigenständiges Programm. Die Kommunikation zwischen dem Portal und GatherYSe geschieht in Form einer Client-Server-Beziehung. Das Portalsystem nimmt hierbei als Client den Dienst von GatherYSe in Anspruch, wobei es ihn lediglich als eine seiner zahlreichen Ressourcen betrachtet. Daher sind beide Teile unabhängig voneinander. So kann sich die Wartung von GatherYSe auf die Erstellung und die Pflege der Anbindungen zu den einzelnen Verlagskollektionen konzentrieren, als auf seine Anpassung an interne Schnittstellenänderungen der Metalib-Software.

Für die Implementierung von GatherYSe wurde auf folgende CPAN-Module zurückgegriffen. Diese sind somit Systemvoraussetzung für die Installation von GatherYSe:

Config::General Syntaxformat für die Provider-Konfigurationsdateien; ähnlich dem Apache-Webserver-Konfigurationsformat.

HTML::TokeParser Verarbeitung von HTML-Code auf der Ebene seiner syntaktischen Grundeinheiten (Starttags mit Attributen, Endtags, Text)

WWW::Mechanize Maschinelle Bedienung von Websites

[4] Application Pragramming Interfaces

[5] Comprehensive Perl Archive Network: http://www.cpan.org/. Ein weltweiter Zusammenschluss von Spiegelservern, von denen jeder die Gesamtheit aller Perl-Ressourcen und Erweiterungsmodule zur Verfügung stellt

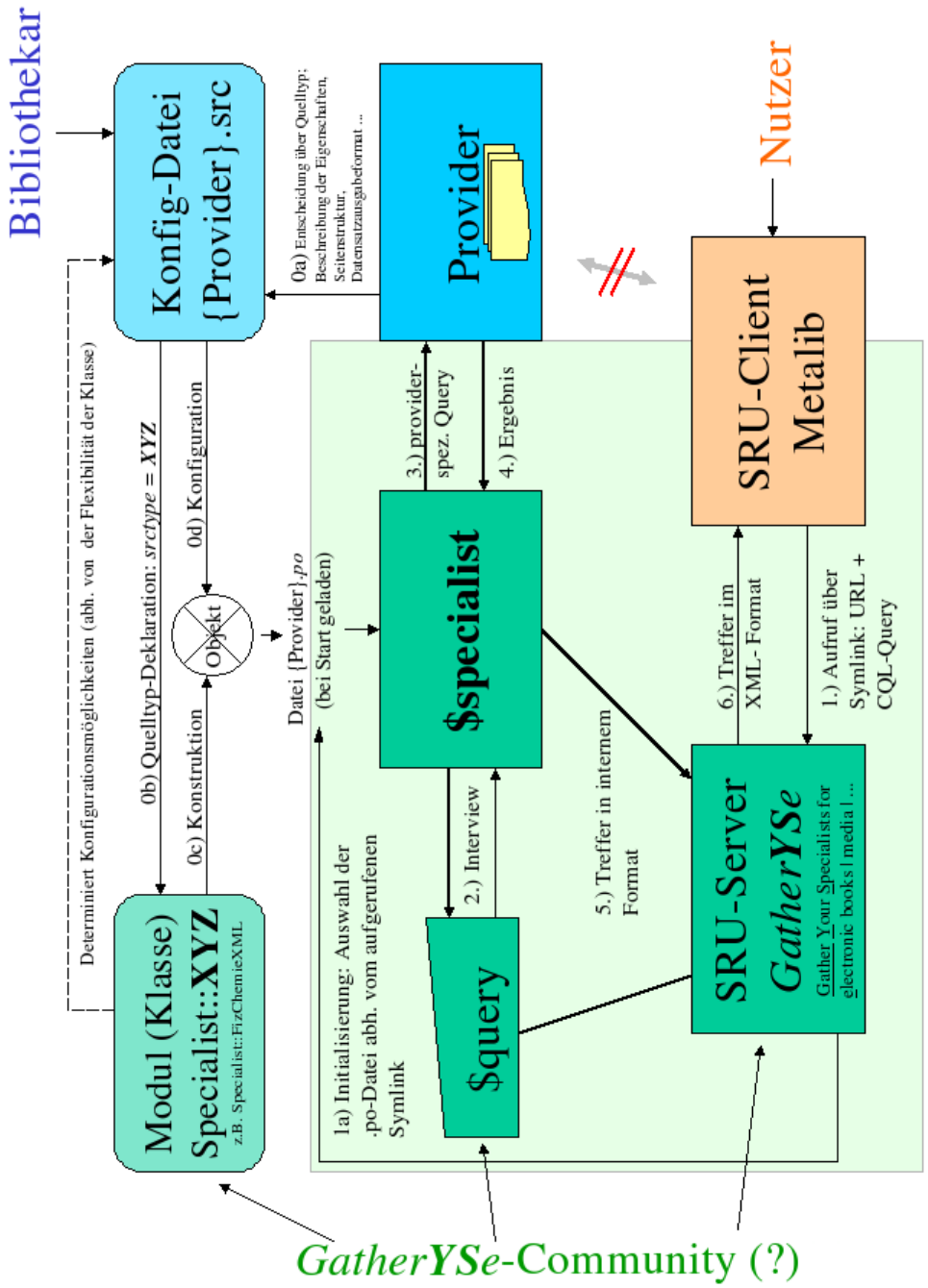

Abbildung 2.1: Architektur der verlagsübergreifenden Suche GatherYSe

SRU::Server für die Kommunikation nach dem SRU-Protokoll aus der Serverperspektive

CGI::Application Webservice-Funktionalität (benötigt durch SRU-Server)

XML::LibXML Verarbeitung von XML

Class::Std::Storable Entwicklung von Klassen nach dem Robustheit fördernden Inside-out-Prinzip [Con06, S. 345ff.], mit der Möglichkeit, generierte Objekte in persistenten Dateien zu speichern und sie aus diesen wiederherzustellen.

Exception::Class komfortables, objektorientiertes Abfangen von Fehlern. Notwendig für das Modul GysDiag (s. 2.5).

Das Protokoll, das für die Kommunikation zwischen Metalib und GatherYSe verwendet wird, ist SRU v1.1 [Lib04], da erstens die Verwendung von standardisierten Protokollen Zukunftstauglichkeit, Beständigkeit und Nachhaltigkeit verspricht, zweitens ein Bundle von SRU-Modulen aus dem CPAN erhältlich ist, und drittens SRU/SRW antwortseitig im Gegensatz zu Z39.50 auf XML und Dublin Core basiert.

Es wurde auch in Betracht gezogen, den SRU-Modus des Net::Z3950-Bundles[6] zu verwenden. Eine Umstellung auf Net::Z3950 in der zukünftigen Entwicklung könnte GatherYSe auch für Systeme zugänglich machen, die SRU noch nicht unterstützen. Diese Alternative wurde jedoch verworfen, da der zeitliche Projektrahmen sowie vor allem der Nutzen mit der Einarbeitung in die komplexe Funktionalität dieser Module in keinem vernünftigen Verhältnis stehen.

Auf eine entsprechende Anfrage des Nutzers hin initialisiert Metalib[7] die SRU-Kommunikation mit dem GatherYSe-Server, und zeigt dem Benutzer wie üblich an: „Suche läuft ...“. An dieser Stelle zeigt Metalib in einer Liste der angefragten Ressourcen den jeweiligen Zustand jeder einzelnen an. Die Ressourcen werden dabei indirekt durch getrennte, parallel auf einem Rechner ausgeführte GatherYSe-Prozesse angesprochen.

Die GatherYSe-Prozess, angestoßen auf eine Anfrage von Metalib bzw. vom Benutzer der E-Book-Sektion von Metalib hin, gliedert sich in mehrere Schritte, die von einzelnen interagierenden Modulen übernommen werden. Leitgedanke war bei der Entwicklung, die Verantwortlichkeiten gleichmäßig zu verteilen und deutlich voneinander abzugrenzen. So wird jede Ressource von einem eigenen Spezialist befragt und ausgewertet. Spezialisten basieren intern auf einer dreistufigen Hierarchie: ihr gemeinsames Verhalten gegenüber dem Hauptprogramm, ihre eigenen statischen, auf einen mehr oder weniger abstrakten Ressourcentyp bezogenen Verarbeitungsprozesse, sowie der aus einer Konfigurationsdatei importierten Ausrichtung auf eine konkrete individuelle Ressource. Das Hauptmodul von GatherYSe ist ausschließlich dafür verantwortlich, die SRU-Anfrage entgegenzunehmen, in der Form eines intelligenten, sich selbst vorinterpretierenden Objekts dem aufgerufenen Spezialisten mitzugeben, und dessen Antwort in das XML-Format zu überführen.

[6] http://search.cpan.org/ sondberg/Net-Z3950-SimpleServer-1.05/

[7] in der aktuellen Version 3 wird dazu ein *external hook* verwendet, s. 4

Außerdem als wichtig erachtet wird die Trennung der Konfiguration und der Benutzung von GatherYSe. Nur immer dann, wenn Änderungen an den Spezialisten bzw. an den Ressourcendefinitionen vorgenommen wurden, sind die Konfigurationsdateien erneut einzulesen (s. Schema, oberen Teil). Wird dagegen GatherYSe vom Nutzer gestartet, braucht lediglich der vorkonfigurierte Code geladen und gestartet zu werden. Zwar ist beim Prototypen die hierdurch gesparte Zeit noch zu vernachlässigen, doch vor allem in Hinblick auf eine mögliche Weiterentwicklung verspricht das Prinzip nach Meinung des Autors hohe Effizienz.

Es werden im Folgenden die einzelnen Schritte sowohl bezüglich ihrer implementierten Funktionsweise als auch Potenziale und Perspektiven für eine mögliche Weiterentwicklung näher erläutert.

2.3.1 Initialisierung

Das zu GatherYSe zugehörige Tool `configure.pl` (s.u.) wird immer dann aufgerufen, wenn die Konfiguration der GatherYSe-Umgebung, insbesondere die Beschreibung der einzelnen E-Book-Anbieter in den src-Dateien verändert wurde. Die Ausführung dieses Skriptes ist oft auch dann erforderlich, wenn bestimmte Änderungen an der Implementierung von Spezialisten (s.u.) vorgenommen wurden. Das Skript erstellt für jeden Anbieter anhand seiner Konfigurationsdatei im Unterverzeichnis `configs/` ein *Specialist*-Objekt und speichert dieses in einer Datei namens *`KurzName`*`.po` im Verzeichnis `init/`. Im konkreten Fall liegen nach dem Durchlauf von `configure.pl` folgende Dateien in diesem Verzeichnis: `netlibrary.po`, `oxfordanb.po`, `oxforddnb.po`, `oxfordref.po` und `oxfordsch.po`. Der Name einer `po`-Datei entspricht der Benennung der jeweiligen `src`-Datei.

Diese Objekte können die Benutzerschnittstelle einer bestimmten Verlagskollektion bedienen, und zwar auf allen betroffenen Problemfeldern:

- Authentifizierung und Initialisierung,
- Zusammenstellung der anbieterspezifischen Abfrage-URL bzw. eines entsprechenden HTTP-Requests anhand des Querys von Metalib,
- Extraktion der Metadaten aus der HTTP-Response (HTML, XML) sowie
- deren Umsetzung in das Format, das letztendlich an Metalib zurück geschickt wird.

In der Prototypversion sind noch alle Spezialisten – außer dem für Springer – an ihre jeweilige Einzelquelle gebunden. Gleich generische, d.h. auf mehrere Quellen eines Typs ausgerichtete Spezialisten zu implementieren hätte eingedenk des engen zeitlichen Rahmens bedeutet, dass die Implementierung lediglich in die Konfigurationsdatei verschoben wird, ohne dass Automatismen und künstliche Intelligenz dem pflegenden Personal Konfigurationsarbeit abnehmen. Nur bei Springer war es naheliegend, bereits einen allgemeinen Spezialisten für den Quelltyp „Ressource via FIZ Chemie Guide“ zu

implementieren, da die Guide-Software offenbar dieselbe für alle Volltextressourcen des FIZ Chemie ist und sie diese allein über eine ID unterscheidet. In Zukunft sollte darauf hingearbeitet werden, die Entwicklung von Spezialisten modular und möglichst einfach zu gestalten, so dass sie einmal auch von weniger Perl- bzw. programmierkundigen Bibliothekaren erstellt werden können. Ein Ansatz in Form eines ausgelagerten, universellen Webscrapers ist beim Prototypen bereits sichtbar (s. 2.4).

Der GatherYSe-Server kann im parallelen oder im seriellen Modus betrieben werden. Für die Integration in Metalib ist ersterer gedacht, um von der Verbundsuchtechnologie des Portalsystems zu profitieren. Dazu wird pro E-Book-Anbieter und Anfrage ein separater GatherYSe-Prozess auf dem Server gestartet. Realisiert wird die Trennung über symbolische Links[8] auf das Skript `run.pl`. Wird einer dieser Symlinks über die URL aufgerufen, stellt das Hauptmodul fest, welcher Symlink dies ist und lädt die entsprechende `.po`-Datei aus dem `init/`-Verzeichnis. In diesem Sinne fungiert das Hauptmodul als ein SRU-Wrapper um den Spezialisten.

Der serielle Modus fragt alle Spezialisten der Reihe nach ab. Er wird aktiviert, indem `run.pl` direkt aufgerufen wird. Da ineffizient, findet dieser Modus praktisch keine Anwendung, kann aber als Rahmen für eine spätere echte, autonome, aktive Parallelisierung dienen, die von GatherYSe kontrolliert und gesteuert wird.

2.3.2 Interpretation der Metalib-Anfrage

Das CPAN-Modul SRU::Server, Bestandteil des SRU-Pakets, nimmt eine Übersetzung der CQL-Syntax (Common Query Language) in einen sog. Parse-tree[9] automatisch vor.

Die Baumstruktur könnte einfach direkt an den Spezialisten übergeben werden. In der Folge müsste jeder Spezialist eigenen Code enthalten, um auf Elementardaten aus der Baumstruktur zuzugreifen und sie zu verarbeiten. Code jedoch, der sich in allen Spezialisten wiederholt, ist schwieriger zu warten, da Änderungen mehrmals vorgenommen werden müssen.

Um dies zu vermeiden, wurde die Baumstruktur in ein Objekt bzw. in einer Klasse namens „IQuery“ (Interviewable Query) ausgegliedert und gekapselt. Das IQuery-Objekt kann von einem Spezialist über eine definierte Methodenschnittstelle ausgelesen werden.

Die Kommunikation zwischen dem Spezialist und dem IQuery wurde dem Auskunftsgespräch zwischen dem Bibliothekar und dem Nutzer nachempfunden: Während der Nutzer – im Ideal – gut über seine Wissenslücke Bescheid weiß, besitzt der Bibliothekar die Kompetenz und Erfahrung über die Retrievalmöglichkeiten und die Bedienung der ihm zur Verfügung stehenden Informationsmittel. So wie es die Aufgabe des Bibliothekars ist, sich der Sprache des Benutzers zu bedienen, um sich ein Bild davon zu machen, nach welchen Informationen dieser sucht, so ruft das Spezialist-Objekt eine

[8] Ein symbolischer Link (kurz: Symlink) ist eine spezielle Dateiart in Unix-Systemen, die den Zugriffspfad auf eine andere Datei repräsentiert. Bei Zugriff auf einen symbolischen Link lädt das Betriebssystem diskret die eigentliche reguläre Datei.

[9] Baumartig verzweigte Anordnung der Querystrukturelemente zur maschinellen Verarbeitung

selbstgewählte, geeignete Auswahl aus den Auslesungsmethoden des IQuery-Objekts in einer Reihenfolge auf, die dem Ablauf der Bedienung der jeweiligen Ressource entspricht.

Das Problem an der Trennung von problemstellungs- und problemlösungsbezogener Verantwortlichkeit ist jedoch, dass es theoretisch unzählige Fragen gibt und ebenso viele Arten der Antwort, die ein Spezialist erwarten kann. Daher besteht längerfristig das Risiko einer zu schnellen, da unbedachten Anreicherung von Auslesungsmethoden als Bestandteil der IQuery-Schnittstelle. Um dem entgegenzuwirken, gibt es die für komplizierte bzw. neuartige Fragestellungen gerichtete provisorische Möglichkeit, von IQuery direkt Zugriff auf die enthaltene Baumstruktur zu erhalten. Dies ist aber mit Bedacht zu handhaben, da damit nicht nur die Beschränkungen, sondern auch die Vorteile von IQuery „umgangen“ werden.

Gegenwärtig unterstützt IQuery lediglich UND-verknüpfte Suchterme, die feldspezifiziert sein können. Außerdem werden zur Zeit nur zwei Relationen (Verhältnis zwischen Feld und seinem Wert) unterstützt: „enthält“ und „exact“. Das prototypische IQuery ignoriert jedoch die unterschiedliche Semantik.

Ebensowenig ist die Trunkierung von Begriffen möglich.

In der weiteren Entwicklung sollte IQuery zunehmend komplexere CQL-Anfragen verarbeiten können und sich gut ergänzende und zusammen wirkende Methoden anbieten, mit deren Hilfe eine Abbildung des Problems in das providerspezifische Format ermöglicht wird. Außerdem könnten in diesem Arbeitsschritt zusätzliche Prozesse zur automatischen oder anwenderbestimmten Anfrageoptimierung stattfinden. Denkbar wäre etwa die Anreicherung mit Synonymen aus Online-Wörterbüchern oder fachlichen Thesauri, oder eine Rechtschreibkontrolle.

2.3.3 Auswertung des IQuery-Objekts

Nach der Konstruktion des IQuery-Objekts, das die ihm übergebene CQL-Anfrage des Clients vorverarbeitet hat, wird die Methode `retrieve()` des Spezialisten mit dem IQuery-Objektreferent als erstem Argument aufgerufen: `$specialist->retrieve( $query )`. Die Methode ist verantwortlich für alle Arbeitsschritte, die zu einer erfolgreichen Sitzung mit dem Informationsprovider notwendig sind, und liefert zum Schluss die Datensätze bereits im Dublin-Core-Format zurück. Das Hauptmodul *GatherYSe.pm* nimmt an ihnen eine Umsetzung in ein SRU-konformes XML-Dokument vor.

Nachdem die Specialist-Methode `retrieve()` ggf. die Sitzung mit dem Informationsprovider initialisiert und vorbereitet hat, gilt es, diesem die Anfrage im erwarteten Format zu senden. Es gibt verschiedene Arten, wie eine Anfrage gesendet wird. Für Daten begrenzten Umfangs wird oft die HTTP-GET-Methode verwendet. Hierbei werden die Elementardaten der Anfrage als eine Kette von `parametername=wert`-Gliedern an den URI[10] angehängt. Zum Beispiel meint das Segment `titlekw=world` in der Anfragezeile an einen hypothetischen Provider, dass „world“ im Titel der gesuchten Medien

[10] Uniform Resource Identifier

vorkommen muss.

Die Anfrage, welche der Spezialist für den Provider aus den Antworten vom IQuery generiert, ist an Vollständigkeit der Resultate orientiert. So wird das Leistungsfähigkeitsgefälle ausgeglichen, das zwischen den Suchmöglichkeiten, die der Provider anbietet, und den Möglichkeiten, die durch die CQL-Sprache bzw. durch Metalib zur Verfügung stehen, besteht. Aspekte der Anfrage, wie zum Beispiel Ausschlüsse mit dem booleschen Operator NOT, die das Suchinterface des Anbieters nicht unterstützt bzw. abbilden kann und die auch nicht auf einem Umweg umsetzbar sind, werden ignoriert, so dass ggf. mehr Ergebnisse als der für den Nutzer relevanten gefunden werden. Später, nämlich im sechsten Schritt, werden diese überschüssigen Ergebnisse herausgefiltert.

Eine Nachfilterung geschieht gegenwärtig nur bei Oxford Reference Online (`Specialist::OxfordRef`).

2.3.4 Verbindung mit dem Anbieter

Die so generierte Anfrage wird nun an den Provider geschickt. Verwendet der Anbieter die HTTP-GET-Methode, erfolgt das Abschicken zum Beispiel durch schlichtes Senden des URIs `http://www.url-des-anbieters.com/such-programm.pl?kriterium1=wert1&kriterium2...` mittels des für die gesamte Interaktion verwendeten CPAN-Moduls WWW::Mechanize über dessen Methode `get( $uri )`.

Bei der HTTP-POST-Methode dagegen müssen die Daten durch WWW::Mechanize, das auch das Ausfüllen von HTML-Formularen beherrscht, in das Suchformular eingetragen werden. Die POST-Methode wird beispielsweise in Oxford Scholarship Online verwendet.

Einschränkend ist jedoch zu erwähnen, dass WWW::Mechanize gegenwärtig und wahrscheinlich auch in Zukunft kein Javascript unterstützt[Les05]. Formulare, deren Funktion vollständig oder nur in essentiellen Teilen durch Javascript realisiert sind, können daher nicht angesprochen werden.

Es ist durch die Trennung von Spezialist und dem GatherYSe-Hauptprogramm möglich, den Spezialisten selbstständig ggf. eine kompliziertere Anfrage in mehrere einfache aufteilen zu lassen, so zum Beispiel, wenn der Anbieter keine Verknüpfung von Suchtermen mit dem OR-Operand ermöglicht, die Anfrage *Term1* AND (*Term2* OR *Term3*) in die Anfragen *Term1* AND *Term2* und *Term1* AND *Term3*. Der Prozess des Abschickens der Anfrage und der Datenextraktion wird dann mehrmals ausgeführt und die einzelnen Treffermengen, durch den Spezialist wieder vereinigt, an das Hauptprogramm gegeben.

2.3.5 Extraktion der Ergebnisse

Die Antwort des Anbieters kommt in HTML-Form zurück, wenn er kein standardisiertes Austauschformat wie XML unterstützt. Dieses HTML muss maschinell verarbeitet werden, vor allem sind die relevanten Daten (Trefferdatensätze) zu extrahieren.

Kompliziert wird dieses Problem durch die Existenz unzähliger Arten und Ausgestaltungen, wie Ergebnisdaten in HTML kodiert werden können, und im Zuge voranschreitender Trennung von Inhalt, Struktur und Darstellung in Webseiten ist es für den Anbieter leicht möglich, die Darstellung der Treffer zu ändern. Sobald dies geschieht, verlöre der jeweilige Spezialist seine Funktion und muss in großen Teilen aufwendig neu geschrieben werden. Um diese Gefahr der Unverlässlichkeit und Unnachhaltigkeit von Spezialisten zu reduzieren, wurde im Rahmen des GatherYSe-Projekts ein Webscraper-Modul namens *HTML::StreamFisher* entwickelt, das patternorientiert die Datenextraktion übernimmt. Das Pattern definiert die HTML-Code-Struktur jedes dargestellten Trefferdatensatzes, sowie auch die Positionen der zu extrahierenden Daten, und kann bei einer anbieterseitigen Änderung vergleichsweise leicht angepasst werden, ohne dass der Perlcode des Spezialisten modifiziert werden braucht.

Zu detaillierten Informationen über HTML::StreamFisher s. Abschnitt 2.4.

2.3.6 Revision, Reduktion und sonstige Nachbehandlung

Die erhaltenen Treffer können sodann auf verschiedene Kriterien hin untersucht werden, die im zweiten Schritt vernachlässigt worden sind. Dieser Schritt soll nachträglich die Ungenauigkeit und Vollständigkeit, worauf die Anfrage an den Provider im zweiten Schritt orientiert war, in Richtung Genauigkeit korrigieren.

Ferner kann hier auf bestimmte Eignungskriterien getestet werden, die nicht im IQuery-Objekt enthalten sind, zum Beispiel ob der Volltext lizenziert und verfügbar ist, wofür diese Unterscheidungen im HTML-Code eindeutig wiedergegeben und auch durch das Pattern berücksichtigt sein müssen.

Auch die Anreicherung von Informationen zu den Treffern aus anderen Quellen ist hier technisch möglich.

2.3.7 Übertragung der Treffer an Metalib

Schließlich werden die Treffer, welche bis jetzt im internen Perlformat vorliegen, in das durch das SRU-Protokoll festgelegte XML/Dublin-Core-Format übertragen. Dazu wird auf die entsprechenden Module aus dem SRU-Bundle zurückgegriffen.

Statt Treffern kann die Antwort diagnostische Informationen über einen Fehler enthalten. Auch hierfür ist den Bestimmungen und Konventionen (Fehlercodes) des SRU-Protokolls zu folgen, so dass der Anwender auf standardisiertem Wege, durch das Bibliotheksportal, über den Grund einer leeren Treffermenge unterrichtet wird. Dies wird gegenwärtig aber weder von Metalib selbst noch vom verwendeten Hook unterstützt. Diagnostische Antworten werden lapidar als „0 Treffer“ fehlinterpretiert.

2.4 Die Auswertung von HTML-Trefferseiten

Für das GatherYSe-Projekt wurde eigens ein Webscraper-Modul entwickelt, das zwar ein integraler Bestandteil von GatherYSe darstellt, aber später auch für andere Anwendungen in Form eines CPAN-Modul geeignet sein könnte.

An dieser Stelle muss betont werden, dass bereits zahlreiche Tools und Module zur Extraktion von Inhalten aus Webseiten im Netz existieren. Einige, namentlich d2c [Vö03], WWW::Scraper, HTML::DOMbo und HTML::ListScraper, wurden daraufhin evaluiert, ob sie für eine Verwendung in Kombination mit den Spezialisten geeignet sind. Aufgrund des engen Zeitrahmens und der gesetzten Prioritäten war es jedoch unmöglich, alle einschlägigen Code-Bibliotheken und Programme auf ihre Eignung zu analysieren und ausreichend zu testen.

Essentielle Anforderungen an das Webscraper-Modul für GatherYSe sind:

1. Unkomplizierte Übertragbarkeit. Bei Hinzunahme einer neu lizenzierten Verlagskollektion an E-books mit eigener Suchoberfläche soll idealerweise eine einfache Beschreibung des Formats der einzelnen Treffer reichen. Auch müssen Anpassungen so einfach sein wie ein Layoutwechsel der Ergebnisseiten für den Provider.

2. Die extrahierten Daten sollen pro Datensatz und Feld in HTML-losen Reinform an den Spezialisten übergeben werden, z.B. in Form von Objekten, die durch ihre Methoden feldweise ausgelesen werden können, oder in Form einfacher Hash-Tabellen. WWW::Scraper kommt dem sehr nahe.

3. Es muss eine Perl-API vorhanden sein.

4. Die Software sollte frei (Open-Source), verfügbar, gut dokumentiert sein, und gegenwärtig eine große Community besitzen.

Die genannten in Betracht gezogenen Exemplare wurden nach eingehender Betrachtung ihrer Dokumentation hinsichtlich der Einhaltung aller Kriterien für nicht geeignet befunden. Das Tool d2c erschien dem Autor als das leistungsfähigste, vor allem durch das WYSIWYG-Konzept, wird aber längst nicht mehr weiterentwickelt. Auch eine öffentlich verfügbare Distribution existiert genauso wenig wie Benutzungshilfen; eine Perl-API ist ebenfalls nicht vorhanden [Vö07].

Deshalb hat sich der Autor zu der Eigenentwicklung entschieden. In seiner Funktionsweise nimmt HTML::StreamFisher Anleihen von folgender Software:

HTML::ListScraper arbeitet auf Basis von Elementsequenzen, ist, beurteilt anhand seiner CPAN-Dokumentation, allerdings weniger leistungsfähig.

WWW::Scraper verwendete externe Definitionen von Suchmaschinen. Scraper-Definitionen erscheinen dabei wie kleine Programme, deren kleinste Einheiten „Op-Codes“ genannt werden. StreamFisher-Patterns arbeiten dagegen beschreibend, d.h. sie sind keine Programme.

Reguläre Ausdrücke nehmen die Mustervergleiche an Text vor. HTML::StreamFisher arbeitet nach einem ganz ähnlichen Prinzip nicht auf Zeichenebene, sondern auf Ebene der HTML-Token[11]. Außerdem von regulären Ausdrücken inspiriert ist der interne Backtracking-Algorithmus, der es erlaubt, zu früheren Prozesspositionen zurück zu kehren.

[11] Syntaktische Einheiten wie Starttag, Endtag und Text

HTML::StreamFisher arbeitet patternbasiert, d.h. der Anwender oder Maintainer beschreibt die Struktur des Datensatzes so, wie er im Quelltext einer Antwortseite des E-book-Anbieters kodiert ist. Diese Patterns arbeiten auf zwei Ebenen: Zum Einen enthalten sie das aus HTML-Tags bestehende Gerüst des Datensatzes, dass die formatierte Darstellung der Daten im Browser ermöglicht. Zum Anderen befinden sich zwischen diesen Tags spezielle, durch umschließende Doppelrauten (`##`) gekennzeichnete Platzhalter, die einen im Pattern eindeutigen Namen besitzen und den eigentlichen Textinhalt symbolisieren.

Das mit einem Pattern initialisierte HTML::StreamFisher-Objekt durchsucht nun den gesamten HTML-Code, dessen Dateihandle oder Referenz-auf-Zeichenfolge an seine Methode process_html() übergeben wurde, nach Strukturen, welche dem Pattern exakt entsprechen. Text im Code, der seiner Position nach einem der Platzhalter im Pattern gegenübersteht, wird jeweils im Datensatz-Hash unter dem Namen des Platzhalters festgehalten.

Es ist möglich, dass die HTML-Strukturen der Datensätze in der Antwort variieren, je nach dem, welche Daten der Datensatz enthält, bzw. um welcher Art von Medium es sich handelt. So kann beispielsweise der Datensatz eines einzelnen E-books kodiert sein mit `<div><h2>Titel</h2>...`, während ein anderer Treffer, der einen Serienteil darstellt, kodiert ist mit `<div><p>Titel der Serie</p><h2>Titel des einzelnen Teils</h2>`. Das Element p ist also nur bedingt vorhanden. Um auch solche Fällen zu berücksichtigen, können Patterns von HTML::StreamFisher verschachtelt werden. Diese Verschachtelung geschieht jedoch nicht „in-place“, wie etwa in mathematischen Termen durch runde Klammern, sondern ein Pattern kann aus mehreren Subpatterns bestehen, die untereinander verlinkt sind. Ein Pattern umfasst mindestens das obligatorische Subpattern „RECROOT“ (record root), das die allen Datensätzen gemeinsame Struktur beschreibt. An jeder Stelle zwischen zwei Tags in einem Subpattern können andere Subpatterns verlinkt werden über die Notation `##[NAME]##`. Die Notationssyntax kennt auch die Möglichkeit der Verkettung (UND-DANN) mit `[...][...]`, der Alternation (ODER) mit Senkrechtstrich `|` innerhalb `[...]`, sowie der Option mit finalem `?` (`[...?]`). Das Pattern für obiges Beispiel ist demnach: `RECROOT => <div>##[SERIESTITLE?]##<h2>##TITLE##</h2>, SERIESTITLE => <p>##SERIESTITLE##</p>`.

Außerdem unterstützt HTML::StreamFisher Perls reguläre Ausdrücke für die Anwendung auf den HTML-Code. So kann beispielsweise die Datenextraktion von HTML-Elementinhalten auf bestimmte Teilstrings beschränkt werden. Werden aus einem Elementinhalt mehrere Teilstrings extrahiert, können diesen separate Feldnamen zugewiesen werden.

Das Pattern für HTML::StreamFisher wird ebenfalls aus der GatherYSe-SRC-Datei eines Providers geladen. Die Pattern-Definition werden von einem `<RecordPattern> ...</RecordPattern>`-Tagpaar eingeschlossen. Ein Beispiel:

```
<RecordPattern>

<variantree>    # variant tree ;-) -- obligatorisch
```

```
RECROOT <<DEF
    <div>##[SERIESTITLE?]##<h2>##TITLE##</h2>
DEF

SERIESTITLE <<DEF
    <p>##SERTITLE##</p>
DEF

</variantree>

<regextract>
# FELDNAME = Regulärer Ausdruck (implizite Flags x, m und s)
TITLE = (.+?) (?: \s - \s (.+?) )?
</regextract>

<distribute>
TITLE = TITLE + SUBTITLE
</distribute>

</RecordPattern>
```

Wird dieses Pattern mit HTML::StreamFisher auf folgenden HTML-Code angewandt:

```
    <div><p>Annual Proceedings on Jokes</p>
        <h2>To have a point, or not to have a point
                - That is here the question</h2>
        <h2>by Humphley Backet</h2>
    </div>
    <div>
        <h2>Joking how-to for free-lancers</h2>
        <h2>by Rose Mary</h2
    </div>

    <div id="pageEnd"><p>Results 1 - 2 of 2</p></div>
```

so erhält man folgende Liste von anonymen Hashes zur weiteren Verarbeitung innerhalb von _retrieve()-Funktion:[12]

```
(
        {
```

[12]Dargestellt ist die Repräsentation eines solchen Hashes in Perl-Syntax

```
            SERTITLE => 'Annual Proceedings on Jokes',
            TITLE => 'To have a point, or not to have a point',
            SUBTITLE => 'That is here the question',
        },
        {
            TITLE => 'Joking how-to for free-lancers',
        },
    )
```

2.5 Fehlerbehandlung

Das SRU-Protokoll[Lib04] definiert in der *Diagnostic*-Sektion über hundert Fehlercodes, die mit standardisierten Meldungstexten verknüpft sind und teilweise mit Detailinformationen versehen werden können (oder müssen). Das zu GatherYSe gehörige Modul `GysDiag` wurde entwickelt, um die durchgängige Verwendung einer SRU-konforme Fehlerbehandlung in den anderen Modulen des Systems zu ermöglichen.

Neben einer speziellen *SRU-Diagnostic Response*, die die geforderten Fehlerinformationen enthält und an den Client geschickt wird, werden über GysDiag gemeldete Fehler zwecks Qualitätskontrolle und Korrektur außerdem in eine Logdatei geschrieben. Dies geschieht über ein Format, das jeweils folgende Informationen aufführt:

1. Datum (Format yyyymmdd),
2. Prozesskennung im Betriebssystem,
3. Fehlerkategorie (`error` Fehler; `debug` Debug-Information / unerwarteter Fehler; `warn` Warnung),
4. SRU-Fehlercode (nicht bei Warnungen),
5. den Query-String in CQL-Syntax,
6. den angesprochenen Spezialist (Kurzname),
7. und schließlich der Meldungstext.

Obwohl hilfreich für die Fehlersuche, wird die Angabe der Anfrage als datenschutzrechtlich problematisch erachtet, da sie in Verbindung mit Informationen aus der Apache-Datei `access_log` Rückschlüsse auf die anfragende Person zulassen kann. Vor dem produktiven Einsatz von GatherYSe sollte gegebenenfalls die Funktion `GysDiag::introduced()` entsprechend modifiziert werden.

Auch in neuen Spezialisten und Programmteilen von GatherYSe, die am Retrievalprozess beteiligt sind, sollte GysDiag unbedingt Verwendung finden:

```
use GysDiag qw(throwDiag);
```

`throwDiag` entspricht in ihrer einfachen Syntax der `die`- oder `Carp::croak`-Funktion. Diese Art der Verwendung wird automatisch als „General System Error“ (Code 1) verstanden:

```
throwDiag "Es ist etwas schiefgegangen";
```

Alternativ kann `throwDiag` mit einer Liste mehrerer, benannter Parameter aufgerufen werden:

- `errno =>` Fehlercode gemäß SRU Diagnostics[13]
- `details =>` Detailinformation wie spezifiziert durch das SRU-Protokoll
- `to_log =>` Die Meldung, die im Logfile erscheinen soll. Wenn weggelassen, erscheint der String in `details` bzw. der String der einfachen Aufrufvariante. Wird der Parameter mit einem definierten unwahren Wert (0|"0) belegt, wird eine Meldung im Logfile unterdrückt.
- `level =>` Der Level, unter dem die Meldung firmiert.[14] Er muss nicht angegeben werden, `GysDiag` wählt je nach Fehlerart aus obigen Möglichkeiten aus.

  ```
  throwDiag errno => 2,
            details => "Wartungsarbeiten; der Server steht bald "
                     . "wieder zur Verfügung."
            ;
  ```

Unerwartete Fehler, die *perl* entdeckt, sowie durch den Entwickler mit `die` oder `Carp::croak` geworfene Ausnahmen werden automatisch mit einem Stacktrace[15] versehen, der SRU-konform ebenfalls an den Client geschickt wird, aber auch die Log-Datei schneller wachsen lässt.

Dies geschieht nicht, wenn stattdessen `throwDiag` verwendet wird. Letzteres ist dann zu empfehlen, wenn die Meldung die Fehlerursache direkt benennt, d.h. der Fehler wahrscheinlich nicht in der Implementierung liegt.

Wird das Backtracing nicht gewünscht, so können die beiden dafür verantwortlichen Zeilen 28 und 36 in `GatherYSe/run.pl` gelöscht oder auskommentiert werden. Seine voreingestellte Unterdrückung bei Verwendung von `throwDiag` ist in der Zeile GysDiag.pm:46 zu ändern. Eine noch feinere Unterscheidung bietet der boolesche `throwDiag`-Parameter `show_trace`.

[13] http://www.loc.gov/standards/sru/diagnostics-list.html – 19.07.2003
[14] siehe Dokumentation zu `Log::StdLog`
[15] Rückverfolgungspfad der Funktionsaufrufe

3 Entwicklung der Spezialisten

Das Gesamtangebot der Humboldt-Universität zu Berlin an lizenzierten E-Book-Kollektionen umfasst zum Zeitpunkt der Erstellung der vorliegenden Arbeit: *MyiLibrary* (Testphase), *Springer Verlag*, *Royal Society of Chemistry*, *Wiley*, *Thomson Gale*, *Knovel Library*, *Oxford University Press* (*Oxford Scholarship Online*, *Oxford Reference Online*, *Oxford Dictionary of National Bibliography*, *American National Biography*), *NetLibrary*, *Beck* und andere über die DFG Nationallizenzen lizenzierte Kollektionen. Für eine eingehende Vorstellung und Analyse dieser Verlagsangebote siehe [Lud07].

Volltextzugriff bieten aus dieser Auswahl lediglich der Springer Verlag (über das Fachinformationszentrum Chemie, Berlin), die Angebote von Oxford und NetLibrary. Aufgrund des eng begrenzten zeitlichen Projektrahmens wurden diese Angebote priorisiert.

3.1 Die Basisklasse aller Spezialisten

Die Basisklasse stellt das gemeinsame Verhalten bereit, das alle Spezialisten im Sinne der Objektorientierung[WCO01] von ihr erben. Das gemeinsame Verhalten bezieht sich vor allem auf vier wesentliche Punkte, die in der angegebenen Reihenfolge von der Methode `retrieve()` durchgeführt werden.

1. Die Überprüfung, ob der Spezialist die gestellte Suchanfrage vollständig verarbeiten kann. Ist dies nicht der Fall, wird eine entsprechende Ausnahme geworfen.

2. Der Aufruf der `_retrieve()`-Methode der abgeleiteten Klasse, die für den providerspezifischen Teil verantwortlich ist. Deren direkter Aufruf von außerhalb der `Specialist`-Klassenhierarchie stellt eine Verletzung der Kapselung dar und führt daher zu einem Programmabbruch.

3. Die Umwandlung der Treffer in kontrolliert auslesbare Datensatz-Objekte. Die einzelnen Spezialisten liefern lediglich eine Struktur von Dublin-Core-Elementen samt ihrer Belegungen zurück. Die Basisklasse wandelt diese Strukturen in `Record`-Objekte[1] um, welche sodann via Objektmethoden vom Hauptprogramm ausgelesen werden. Außerdem wird allen Datensätzen der Verlagsname sowie die Typauszeichnung „Text“ (encoding scheme `DCMIType`) hinzugefügt.

4. Die Beurteilung der Retrievalergebnisse. Gegenwärtig wird nur getestet, ob die Ergebnismenge das global für alle definierte bzw. das für einzelne Spezialisten

[1] Die Definition der `Record`-Klasse befindet sich in derselben Datei unterhalb der `Specialist`-Klasse

eingestellte Limit übersteigt (Check `moreAvailable`), und ob der jeweilige Spezialist alle vom Provider als geliefert deklarierten Ergebnisse verarbeitet hat (Check `someSkipped`). Außerdem gibt es einen Test für Spezialisten, die eine Nachfilterung der Ergebnisse vornehmen (Check `reducedSet`).
Gegebenenfalls wird der Treffermenge ein zusätzlicher Datensatz („Appendix“) angehängt, der die fehlenden Treffer numerisch ausweist und einen Link auf die Originalseite des Providers enthält. Hierbei entscheidet sie sich zwischen der ersten und der nächsten, vom Spezialist nicht mehr verarbeiteten Ergebnisseite. Auf keinen Fall werden mehrere Appendizes an die Treffermenge gehängt. Fallen mehrere Checks positiv aus, belegen ihre Hinweise mit `++` verkettet dasselbe Feld `dc.description` in einem Appendix.

Letzteres lässt sich über die Datei `configs/defaults.rc` oder für den einzelnen Spezialisten in dessen `.src`-Datei wie folgt abstellen oder einschränken:

```
appendix yes    # Appendix für alle Checks
appendix no     # Appendix für keinen Check

appendix if moreAvailable, ...  # Appendix nur für bestimmte Checks
```

3.2 Allgemeiner Aufbau

Alle Spezialisten müssen folgende fünf Methoden zur Verfügung stellen:

configure() verarbeitet im Rahmen eines `configure.pl`-Laufs jene Werte aus einer `.src`-Datei, die *nicht* in der Datei `defaults.rc` stehen können, da ihre Angabe quellenspezifisch ist. Die Datei `defaults.rc` hat folgenden Inhalt:

```
# ACHTUNG: Keine Unterstützung von Blöcken und multiplen Vorkommen

results-limit 100   # Änderung problematisch (s. Kommentare in
                      springer.src, oxfordanb.src)

appendix yes
```

what_is_supported() gibt eine Referenz auf eine Hash-Datenstruktur zurück, welche die unterstützten Suchfunktionen beschreibt. Gegenwärtig muss diese nur die Elemente `FIELDS` und `BooleanAND` enthalten. Ersteres muss eine Liste von Dublin-Core-Indizes sein, auf die Suchterme qualifiziert sein können. Zweiteres muss einen wahren Wert erhalten, denn die Unterstützung von boolescher UND-Verknüpfung ist eine Mindestanforderung an die Suchschnittstelle des Anbieters.

_retrieve() ist verantwortlich für die Bedienung der Anbieter-Website und die Extraktion der Ergebnisse. Diese müssen sodann von `record2dc()` verarbeitet werden, bevor eine Referenz auf die Liste der Dublin-Core-konformen Datenstrukturen zurückgegeben wird.

record2dc() nimmt die Umformung der Inhalte eines Datensatzes ins Dublin-Core-Format vor, wobei jedoch die Vereinbarung zwischen den technischen Empfehlungen der Initiative [PJ03], den hart kodierten Erwartungen des verwendeten *external hooks* (s. 4), sowie auch der Funktionalität des SRU-Bundles der aktuellen Version teilweise Kompromisse erforderte. So empfiehlt die DCMI, *element refinements* bzw. *encoding schemes* mit dem `xsi:type`-Attribut auszudrücken. Dies wurde nicht umgesetzt, da das SRU-Bundle der aktuellen Version keine Namespace-Deklarationen unterstützt. Encoding schemes werden daher provisorisch mit dem `scheme`-Attribut ausgedrückt. Auch erwartet der external hook einen Hyperlink im Element `identifier.uri`, während die DCMI die Notation `dc:identifier scheme="dcterms:URI"` empfiehlt (ebd. note 1).
Gibt die Referenz auf eine Hash-Datenstruktur zurück, die die Elemente mit ihren Belegungen enthält. Zusätzlich kann sie ein Element `ATTR` enthalten, das in Form eines untergeordneten Hashs datensatz- oder sogar elementspezifische Attribute definiert.

get_dc_attributes() gibt die Referenz auf eine Hash-Datenstruktur von Feldattributen und deren Belegungen zurück, die alle Datensätze eines Spezialisten auszeichnen. So kann beispielsweise das Format des `date`-Elements aller Datensätze definiert werden.

3.3 NetLibrary (netlibrary)

3.3.1 Nutzungsbedingungen

NetLibrary verbietet den Einsatz von Webscrapern ausdrücklich in ihren Terms of Use:

> Furthermore, you agree not to: (i) use or attempt to use another's account, password, service, or system without authorization from NetLibrary; (ii) access or attempt to access any Content which you are not authorized to access; or (iii) try to reverse engineer, reverse assemble, reverse compile, decompile, disassemble, or otherwise alter any executable code, contents, or materials on or received via this site. You understand that such actions are likely to subject you to serious civil and criminal legal penalties and that NetLibrary will act to protect its rights and the rights of publishers, authors, and its other licensors. Without the express written permission of NetLibrary, you agree not to use any automated data or content gathering and extraction methods, including data mining and robots, in connection with the site.[2]

Um NetLibrary dennoch in der verlagsübergreifenden Suche berücksichtigen zu können, wurde NetLibrary von der Universität um eine Sondererlaubnis gefragt.
Diese wurde erteilt.

[2] http://www.netlibrary.com/TermsOfUse.aspx – 12.06.2007

3.3.2 Konfiguration

Die Datei `configs/netlibrary.src` hat folgenden Inhalt:

```
name = OCLC NetLibrary
srctype = NetLibrary
base = http://www.netlibrary.com/
results-limit 45    # Defaultwert 100 zu langsam

<RecordPattern>

<variantree>

RECROOT <<DEF
<table border="0" width="100%">
<tr>
<td><span>##NUMBER##</span></td>
<td></td><td>
<div>
<span class="relevance" style="##RELEVANCE##"></div>
<span>
<strong>##TITLE##</strong>
<div class="resultsMetadata">
##AUTHORPUBLICATION##
</div>
<div class="summary">
##SUMMARY##
</div>
<span><a href="##EBOOK_URL##">
DEF

</variantree>

<regextract>
RELEVANCE width\: (\d+.\d+%) \;
AUTHORPUBLICATION (?: by \s (.+?) \s+ )? Publication: \s (.+)
</regextract>

<distribute>
AUTHORPUBLICATION = AUTHORS + PUBLICATION
</distribute>

</RecordPattern>
```

3.3.3 Ablauf

Die Generierung und Sendung einer Anfrage an NetLibrary ist aufgrund der Javascripte auf der Seite der Suchmaske, welche die Formularfunktionalitäten erst direkt vor dem Abschicken dynamisch einbauen, eine instabile Angelegenheit. Da WWW ::Mechanize kein Javascript verarbeiten kann [Les05], musste die Semantik des sichtbaren Codes nach Perl in den Spezialisten übertragen werden. Javaskripte, welche in die Seite nur eingelinkt, jedoch nicht eingebettet sind, konnten nicht eingesehen werden. So können bedingt erforderliche Umwandlungen, die jener Code eventuell vornimmt, nicht stattfinden. Entsprechende Suchanfragen des Spezialisten in einer nicht erwarteten Form führen zu einem serverseitigen Fehler.

Zu WWW::Mechanize gibt es keine annähernd praktikable, jedoch zusätzlich javascript-fähige Alternative. Eine javascriptfähige Lösung ist Selenium [Ope], die jedoch einen ganz verschiedenen Zweck verfolgt: das automatische Testen von Webapplikationen. Deren zweckentfremdeter Einsatz für die Ansteuerung der NetLibrary-Suchmaske würde eine eigenständige Browser-Instanz erfordern, welche bei jeder Anfrage erneut geladen werden müsste. Außerdem erforderlich ist ein persistent aktiver Java-Proxyserver, der zwischen dem Browser, dem zu testenden Code und Selenium vermittelt. Aufgrund des vermuteten inakzeptablen Kapazitätsverbrauchs und da erste Tests mit der Javascript-Umgehungslösung Erfolg zeigten, wurde diese Lösung nicht weiter verfolgt.

Zu Beginn wird das WWW::Mechanize-Objekt konstruiert (Z72). Die `autocheck`-Option bewirkt, dass Fehler auf Ebene des HTTP-Protokolls automatisch erkannt werden. Um Fehler, die aus browserspezifischem Seitenverhalten hervorgehen auszuschließen, gibt sich das Objekt als ein Mozilla-Browser aus (Z74), da ein solcher auch für die manuelle Seitenanalyse verwendet wurde.

Die Homepage muss bereits hier aufgerufen werden, um eine valide Sitzungs-ID zu erhalten. (Z79f.)

Die Suchmaske ist laut HTML-Code zwar ein POST-Formular, nach der Verarbeitung durch das Javascript wird jedoch eine GET-Anfrage an das serverseitige Programm `SearchResults.aspx` geschickt.

Jedoch muss trotzdem die Suchmaske geladen werden (Z82-116), da sie für die Anfrage notwendige, zu extrahierende Parameter enthält. Hierbei handelt es sich um die Parameter `__EVENTTARGET`, `__EVENTARGUMENT`, `VIEW_STATE_FIELD_NAME` und `__VIEWSTATE`, von denen allerdings nur `VIEW_STATE_FIELD_NAME` tatsächlich mit einem Wert belegt ist. Dieser kann weder weggelassen noch willkürlich gesetzt werden, ohne dass NetLibrary mit einer Fehlerseite antwortet.

Daneben sind außerdem noch folgende Parameter zu belegen (Z87ff.): `ui` mit „adv“, `db_ebks` mit „on“ (Schalter, welcher die E-books mit einbezieht), `booleanButton` mit „Search“, `ql` mit „ENG“ (query language), sowie `so` mit „Relevance“.

Die eigentlichen Suchfelder sind im Seitenquellcode, wie alle anderen vorhandenen Felder ebenfalls, nur durch ihre *id*-Attribute identifiziert. Sie bekommen ihre für die Formulardatenverarbeitung erforderlichen *name*-Attribute erst durch das eingebettete Javascript, welches vom Browser im Rahmen des Submit-Ereignisses ausgeführt wird.

Die tatsächlich vom Server empfangene Anfrage enthält für jedes der vier Suchfelder folgende Parameter:

tN ist der vom Benutzer eingegebene Wert,

ttN ist der Name des Feldes, das mit diesem Wert belegt ist, sowie

toN ist der zugehörige boolesche Operator.

N ist hier die Nummerierung 1 bis 4. Die Suche wurde jedoch mit höheren Ziffern ebenso erfolgreich getestet.

Sind die Suchterm-Parameter belegt, wird der URL zusammengesetzt (Z155ff.) und abgeschickt (Z163). Diese URL wird gespeichert und für eventuelle spätere Verwendung deklariert[3], und die Antwort auf Erfolg überprüft (Z165-172).

Daraufhin werden aus der erhaltenen ersten Ergebnisseite die Gesamtzahl der Ergebnisse sowie die Anzahl der gelieferten Ergebnisse extrahiert. Mit diesen Zahlen wird der Zustand des Spezialisten abermals aktualisiert. (Z179-192).

Das mit dem Spezialisten verbundene HTML::StreamFisher-Objekt extrahiert nun die Trefferdaten aus der Ergebnisseite (Z203f.).

Schließlich erfolgt ein Sprung auf die gegebenenfalls vorhandene nächste Ergebnisseite. Ist das in der Konfiguration bzw. der `defaults.rc`-Datei definierte Limit jedoch erreicht, wird lediglich die URL auf die nächste Seite gespeichert, damit sie vom Nutzer aus dem Appendix manuell aufgerufen werden kann. (Z206-224)

3.3.4 Feldbelegungen

Das Dublin-Core-Feld `title` wird mit dem Inhalt an der Position `TITLE` (s. Pattern) belegt (Z244). `AUTHORS`, das auch leer sein kann, wird in ein oder mehrere `creator`-Elemente übertragen (Z246). `SUMMARY` wird nach `description` gemappt (Z248). Die Position `PUBLICATION` ist oft zweiteilig: Neben dem `publisher` kann auch ein Jahr angegeben sein, welches `date` zugeordnet werden muss (Z250-254). Dieses Feld wird für alle NetLibrary-Sätze mit dem *encoding scheme* „YYYY“ belegt. [WW98] Der Volltext-Link für das Feld `identifier.uri` wird aus der Position `EBOOK_URL` geholt und muss in eine absolute URL umgewandelt werden (Z256-261).

3.3.5 Diagnostik

„Request for netlibrary.com homepage failed“: Die Homepage konnte nicht geladen werden. Es ist zu überprüfen, ob sich die Domain von NetLibrary geändert hat.(Z80)

„Request for netlibrary advanced search mask failed“: Die erweiterte Suche konnte nicht aufgerufen werden. Die Richtigkeit des URL-Zusatzes ist zu überprüfen. (Z85)

[3]Solche Deklarationen dienen der ständigen Bekanntgabe des aktuellen Zustands

„No value for $property“: Für eins der oben gelisteten versteckten Formular-Attribute konnte nicht der Wert ermittelt werden. (113)

„Couldn't load results page“ Die Antwort des Anbieters auf die Anfrage-URL war nicht erfolgreich. (Z172)

„Odd content of response“: Die Antwort wurde nicht als Trefferseite erkannt. (Z201)

„Can't find any data to extract“: Die Antwort wurde zwar als Ergebnisseite erkannt, doch HTML::StreamFisher konnte keine Daten finden. Dieser Fehler weist darauf hin, dass die Pattern-Definition überholt werden muss. (Z204)

„Can't find next-link“: Obwohl die Zahl der bisher gelieferten Treffer noch nicht die Gesamtzahl erreicht hat, kann kein Link auf die nächste Ergebnisseite gefunden werden. (Z210)

„Couldn't get next page“: Die nächste Ergebnisseite konnte aus anderen Gründen nicht angesprungen werden. (Z216)

3.3.6 Testerfahrungen

Specialist::NetLibrary wurde mit dem Einzelstichwort „world“ getestet. Von den 105 Treffern, welche der Provider in 7 Seiten à 15 Treffern geschickt hat, wurden jedoch nur 84 erkannt. Dies ist sehr wahrscheinlich auf nicht berücksichtigte Varianten in der Quellkodierung und/oder überstrengen regulären Ausdrücken im Block `RecordPattern/regextract` in der NetLibrary-Konfigurationsdatei zurückzuführen.

Bei einem Limit von 45 Treffern tritt dieser Ausfall lt. Appendix nicht auf. Er weist nur die Gesamttrefferanzahl von 3712 Stück aus. Die Suche nach „world“ und „autrans“ ergab nur 2 Treffer, es wurde kein Appendix angehängt. Ein auf `title` qualifizierter Term „sarajevo“ gab ein Ergebnis zurück. Ohne Qualifizierung sind es 47; 45 wurden geholt. Aus dieser Menge wurde ein Treffer ausgewählt, der „canada“ im Titel enthielt, und eine kombinierte Suche mit unqualifiziertem „sarajevo“ und auf `title` qualifiziertem „canada“ kam erwartungsgemäß dieser eine Treffer zurück.

Doch die Phrasensuche funktionierte nicht wie erwartet. Die Suche nach „critisms were made“ ergab 43 Treffer, von denen ein sehr großer Teil ein KWIC enthielt, das zwar meist „critisms“ und „made“ nahe beieinander, jedoch nicht die exakte Phrase beinhaltete. Es wird vermutet, dass anbieterseitige Stoppwortalgorithmen für dieses Verhalten verantwortlich sind.

3.4 American National Biography (oxfordanb)

3.4.1 Nutzungsbedingungen

Die Nutzungsbedingungen der ANB gelten auch für die anderen Oxford-Kollektionen.

> Under the license signed by the Subscriber, you may NOT: [...] 1. remove or alter the copyright notices or other means of identification or disclaimers as they appear in American National Biography Online;
>
> 2. systematically make printed or electronic copies of multiple extracts of American National Biography Online for any purpose;
>
> 3. display or distribute any part of American National Biography Online on any electronic network, including without limitation the Internet and the World Wide Web (other than the institution's secure network, where the Subscriber is an institution);
>
> 4. permit anyone to access or use American National Biography Online (other than other users authorized by the institution, where the Subscriber is an institution);[4]

Absatz 1 trifft insofern nicht zu, dass zum Einen der Zugriff auf die verlagsübergreifende Suche durch Metalib über einen vorgeschalteten Disclaimer führt, der dem Nutzer populäre Verwendungsarten, die für die Anbieter nachteilhaft sind, ausdrücklich verbietet. Zum Anderen werden die Trefferdatensätze deutlich mit dem Namen des Anbieters versehen. Eine direkte Manipulation an entsprechenden Hinweisen findet nicht statt. Der Zugriff auf ANB erfolgt indirekt, aber transparent durch GatherYSe, weshalb die Hinweise eher aufwendig in die Metalib-Umgebung integriert werden müssten.

Die Problematik des zweiten Absatzes hängt im Wesentlichen ab von der Auslegung des Begriffs „systematisch". Nach Interpretation des Autors impliziert dieser Begriff die Verwendung einer Datenbank, in welcher die extrahierten Inhalte persistent und unabhängig vom Anbieter gespeichert und weiterverbreitet werden könnten. Eine solche Datenbank ist jedoch nicht vorhanden. Die Datenextraktion durch GatherYSe bzw. durch das Modul HTML::StreamFisher geschieht auf eine dedizierte individuelle Anfrage eines Benutzers auf dem Campus hin und erfolgt mit dem ausschließlichen Ziel der Darstellung der Ergebnisdaten innerhalb Metalib.

Die Absätze 3 und 4 sind mit einer Einschränkung in Klammern versehen, woraus implizit eine Erlaubnis des Projekts herausgelesen werden kann, soweit der Zugriff auf die Oxford-Produkte ausschließlich Nutzern auf dem Campus gewährt wird.

Es handelt sich hierbei ausdrücklich um einen Laienkommentar. Bevor GatherYSe für die alltägliche Verwendung durch die Nutzer der Universitätsbibliothek freigegeben wird, wird empfohlen, die Angelegenheit juristisch überprüfen zu lassen.

3.4.2 Konfiguration

Die Datei `configs/oxfordanb.src` hat folgenden Inhalt:

```
name = American National Biography
srctype = OxfordANB
```

[4] https://www.anb.org/support.html – 12.06.2006 14:20

```
base = http://www.anb.org/
# results-limit problem at RecordPattern/regextract

<RecordPattern>

<variantree>

RECROOT <<DEF
        <li>
        <a href="##ARTICLE_URL##">
        ##FULLNAME##
        </a>
        ##[DESCRIPTION][IMAGE?]##
        </li>
DEF

DESCRIPTION <<DEF
    ##DESCRIPTION##
DEF

IMAGE <<DEF
<image>##[SCNDIMG|DESCRIPTION?]##
DEF

SCNDIMG <<DEF
        <image>##DESCRIPTION##
DEF

</variantree>

<regextract>
# exclude right column "Background articles". Recommended especially
# when results-limit is higher than one page, because this column is
# displayed with the same entries (first 10) every next page, too.
# DESCRIPTION = \A [^,]
</regextract>

<distribute />

</RecordPattern>
```

Über die Einstellung *srctype* wurde der Anbieter an den Spezialisten „Specialist::OxfordANB“ gebunden. Der Konfigurationsblock „RecordPattern“ definiert die HTML-Struktur der Datensatzdarstellung der Trefferseiten, die so durch HTML::StreamFisher

verarbeitet werden können.

Im Pattern musste eine besondere Behandlung das <image>-Tag erfahren. Das normalerweise für Bilder verwendete „phrase-level“ HTML-Tag <img> würde dagegen automatisch übergangen.

3.4.3 Ablauf

Die Methode `_retrieve()` beginnt mit der Konstruktion eines WWW::Mechanize-Objekts (Z68), bei dem die `autocheck`-Option aktiv ist und somit manuelle Prüfungen des Erfolgs einer Sendung auf Ebene des HTTP-Protokolls nicht mehr notwendig sind. Das Objekt weist sich als ein Mozilla-Browser aus (Z70), da ein solcher auch bei der der Implementierung vorhergehenden Seitenanalyse verwendet wurde. Die Hauptaufgabe der Initialisierungsphase besteht jedoch in der Belegung einiger Default-Parameter (Z76ff.), welche auch bei einer manuellen HTTP-GET-Abfrage in der Adresszeile erscheinen: Über den Parameter `meta-dc` wird die anzuzeigende Treffermenge pro Seite auf 100 Treffer gestellt. Ferner wird der Parameter `func` auf „advanced_search“, sowie `realms_top` auf „General+Areas“ gesetzt, außerdem `idxa` mit „-at“ und `idxb` auf „-bib“. Letztere beide sind offenbar Indexparameter für interne Funktionen des Providers.

Die Abfrage gestaltet sich einfach. Es gilt schlicht, die URL mit den notwendigen Formulardaten zusammenzustellen und sie nach der GET-Methode abzusenden. Volltext-Stichworte kommen in das Feld `fulltext`, Personennamen in das Feld `fieldName`. (Z86-96)

Der Spezialist erwartet sodann die erste Ergebnisseite. Nachdem er über die `_update()`-Methode ihre URL deklariert hat, überprüft er den HTTP-Ergebniscode auf Erfolg (Z110f.). Anschließend werden aus der Ergebnisseite zuerst die numerischen Daten zur Treffermenge extrahiert und entsprechend der Zustand des Spezialisten via `_update()` erneut aktualisiert (Z118-141).

Dann erfolgt die eigentliche Datenextraktion durch das mit dem Spezialisten aggregierte HTML::StreamFisher-Objekt.

Abhängig davon, ob die Anzahl der gelieferten Ergebnisse noch nicht die Gesamtzahl erreicht hat, wird die nächste Ergebnisseite angesprungen (Z150-158). Hat sie jedoch das mit `results-limit` in der Konfigurations- oder der `defaults.rc-Datei` eingestellte Limit erreicht, wird in diesem Fall nur die URL auf die nächste Ergebnisseite in den Zuständen gespeichert (Z168), damit sie im Appendix erscheint.

3.4.4 Feldbelegungen

Das Dublin-Core-Element `title` wird mit dem `FULLNAME`, `description` mit `DESCRIPTION` belegt. (Z182f.) `identifier.uri` wird mit `ARTICLE_URL` belegt. Diese URL ist relativ, musste also um den Basisteil erweitert werden. [Zeile 177ff.] Das Feld `format` erhält den Wert „text/html“.

3.4.5 Diagnostik

Die nachfolgenden Meldungen erscheinen in Form einer SRU Diagnostic Response als „General system error“ (Code 1). Das Portalsystem wird lediglich „0 Treffer“ melden (s. 2.5). Allein über das Testinterface (3.11) sind diese Fehler als solche zu erkennen.

„Couldn't load results page“: Die Antwort des Anbieters auf die Anfrage-URL war nicht erfolgreich. (Z110f.)

„Can't find a results span“: Es wurde zwar eine Treffergesamtzahl, aber keine Von-bis-Spanne entdeckt. (Z128f.)

„Odd content of response“: Die Antwort wurde nicht als Trefferseite erkannt. (Z147)

„Can't find any data to extract“: Die Antwort wurde zwar als Ergebnisseite erkannt, doch HTML::StreamFisher konnte keine Daten finden. Dieser Fehler weist darauf hin, dass die Pattern-Definition überholt werden muss. (Z149)

„Can't find next-link“: Obwohl die Zahl der bisher gelieferten Treffer noch nicht die Gesamtzahl erreicht hat, kann kein Link auf die nächste Ergebnisseite gefunden werden. (Z154ff.)

„Couldn't get next page“: Die nächste Ergebnisseite konnte aus anderen Gründen nicht angesprungen werden. (Z161)

3.4.6 Testerfahrungen

Die Suche nach dem Wort „criticism“ ergab lt. Appendix insgesamt 1726 Treffer. Er behauptet, dass aufgrund des Limits nur 100 Treffer geholt wurden, da die Treffer in der Randspalte hierbei nicht mitgezählt werden.

Die Mehrwortsuche verlief nur teilweise erfolgreich. Während die Suche nach „criticism“ und „miller“ 2 Treffer lieferte, warf die Suche nach „criticism“ und „logical“ einen General System Error („Odd content of response“). Offensichtlich wurde es bei diesem Spezialist versäumt, den 0-Treffer-Fall zu berücksichtigen (Z141). Er sollte hinzugefügt werden, wenn auch auf der Metalib-Oberfläche auf Grund einer Unzulänglichkeit des external hooks „0 Treffer“ erscheint (s. 2.5).

Dieser Fehler wurde erfolgreich korrigiert. Es wurde ferner festgestellt, dass die UND-verknüpfte Suche tatsächlich eine Phrasensuche war. Auch dies wurde berichtigt, so dass auf die Suchanfrage „criticism“ und „miller“ nunmehr 125 Treffer geliefert werden.

Die Suche nach „harrison“ im `title` ergab 55 Treffer.

Eine Suche nach unqualifiziertem „criticism“ und auf `title` beschränktem „harrison“ ergab 16 Treffer, von denen jedoch nur die ersten 6 „harrison“ im Titel enthalten. Der Rest stammt aus der Randspalte.

Die Phrasensuche nach „world peace“ ergab 106 Treffer, d.h. 100 von 125 biografische Artikel zur Suchfrage und 6 Hintergrundartikel aus der Randspalte.

3.5 Oxford Dictionary of National Biography (oxforddnb)

3.5.1 Nutzungsbedingungen

Der Wortlaut der DNB-Nutzungsbedingungen entspricht denen der ANB.[5]

3.5.2 Konfiguration

Die Datei `configs/oxforddnb.src` hat folgenden Inhalt:

```
name = Oxford Dictionary of National Biography
srctype = OxfordDNB
base = http://www.oxforddnb.com/

recognizeResultsFrame = results-subjectslist

<RecordPattern>

<variantree>

RECROOT <<DEF
<li>
<a href="##ENTRY_URL##">
##FULLNAME##
</a>
##DESCRIPTION##
</li>
DEF

</variantree>

<regextract>
DESCRIPTION (.+?) (?: $ | \s(?=\.{3}) (.+))
# last capture (.+?) comes to SUBTEXT and is empty when PEOPLE search
# was used
</regextract>

<distribute>
DESCRIPTION = BIOGRAPHY + SUBTEXT
</distribute>

</RecordPattern>
```

[5] http://www.oup.com/oxforddnb/legal/ – letzter Zugriff: 12.06.2007

3.5.3 Ablauf

OxfordDNB stellt mehrere Suchmasken zur Verfügung (vgl. [Lud07, S. 43]). Relevant im Rahmen des GatherYSe-Projekts ist nur die Suche nach Personennamen und Volltextstichwörtern. Die Suchmaske *People* stellt neben einigen biografischen Suchfeldern auch ein Volltextfeld zur Verfügung, weshalb die Maske *Fulltext* – eingedenk der durch die prototypische IQuery-Klasse eingeschränkten Nutzbarkeit ihrer erweiterten Funktionen – nicht zu verwendet werden braucht.

Auch dieser Spezialist beginnt mit der Erstellung eines WWW::Mechanize-Objekts mit der Option `autocheck` und der Identifikation als Mozilla-Browser (Z74-76). Um sich beim Anbieter zu authentifizieren, wird nun die URL `http://www.oxforddnb.com/subscribed/` aufgerufen (Z80f.).

Sodann werden die Default-Parameter belegt, wie sie in der URL der Ergebnisseiten bei einer manuellen Suche erscheinen[6]: `display` mit 100 (Treffer pro Seite), `search` mit „Search“, `textQualifier` mit „EXACT“[7] sowie `textFieldLimiter` mit „article_text“. (Z92ff.)

Anschließend erfolgt die Zusammenstellung des Querys (Z101-105). Das Feld `simpleName` wird mit „+“ verketteten `title`-Suchtermen gespeist, `text` mit unqualifizierten Suchtermen. In beiden Fällen werden die Suchterme mit Anführungszeichen versehen, um Phrasensuche zu ermöglichen. So wird die URL zusammengestellt und abgeschickt (Z107-116).

Als Antwort wird ein HTML-Frameset erwartet. Oben befinden sich Informationen zur Ergebnismenge sowie Navigationsfunktionen. Im unteren Frame befindet sich die Trefferliste.

Eine zusätzliche Herausforderung ergibt sich dadurch, dass bei einer Suchanfrage, die einen einzelnen Treffer ergibt, dieser automatisch angewählt wird, ohne dass eine Trefferliste erscheint. In diesem Fall muss im unteren Frame Volltext erkannt werden (Z137ff.) und aus ihm geeignete Textstellen als Metadaten extrahiert werden: Das Dublin-Core-Element `title` wird mit dem extrahierten Namen der Person, `description` mit den dahinterstehenden geklammerten Lebensdaten belegt. Mit dem anschließend um die sitzungsunabhängige Query-URL und eine Formatangabe erweiterten Datensatz kehrt die Methode frühzeitig zurück (Z148ff.).

Wird der Inhalt des unteren Frames nicht als Einzel-Artikel erkannt, geht der Spezialist von einer Trefferliste aus. In diesem Fall extrahiert er die numerischen Angaben zur Ergebnismenge (Z173ff., Z196f.) und aktualisiert entsprechend seinen Zustand (Z180ff., Z201). Im Falle eines providerseitigen „0-Treffer“-Ergebnisses kehrt die Methode ohne Rückgabewert zurück (Z188).

Abhängig von der Anzahl der bisher erhaltenen Ergebnisse gegenüber der Gesamtanzahl und des eingestellten Limits wird die nächste Ergebnisseite angewählt (Z206-Z229). Da sich jedoch der Link auf die nächste Seite der Trefferliste im oberen, nicht geladenen Frame befindet, und ihre URL überdies an die Sitzung gebunden ist, muss ihre sitzungsabhängige URL direkt generiert werden (Z208ff.). Diese URL wird abge-

[6] Die darin angebenen dynamischen Werte Werte haben sich als nicht notwendig erwiesen

[7] Es wird vermutet, dass dies Ganzwortsuche bedeutet, nicht Phrasensuche.

schickt und die Antwort jeweils auf Erfolg getestet, ehe sie in der folgenden Iteration verarbeitet wird. Wird das eingestellte Limit überschritten, wird die URL in eine sitzungsunabhängige Form gebracht, die der Nutzer aus dem Appendix anwählen kann (Z233-239).

Die erhaltenen Ergebnisse werden schließlich nach dem Dublin-Core-Standard konvertiert als Referenz auf sie zurückgegeben (249f.).

3.5.4 Feldbelegungen

Es werden die Dublin-Core-Elemente `title` aus der Position `FULLNAME` und `description` aus `BIOGRAPHY` und `SUBTEXT` gespeist. Diese kommen in getrennte Felder des gleichen Elements, da zwischen ihnen kein direkter semantischer Zusammenhang besteht.

Das Element `identifier.uri` aus mit der Position `ENTRY_URL` belegt. Die relative URL wird hierfür in die absolute Form gebracht.

Weder in der URL, noch an anderer Stelle im Quelltext eines Treffers wird ein Digital Object Identifier angegeben.

Das `format` wird mit „text/html“ angegeben.

3.5.5 Diagnostik

„Could not authenticate to provider“: Es konnte nicht die Athentisierungs-URL aufgerufen werden (Z80ff.).

„Search raised a server error“: Die Antwort auf die Query-URL ist nicht erfolgreich (Z118f.)

„Getting single page raised provider server error“: Die Anfrage der Einzeltrefferseite wurde nicht erfolgreich beantwortet (Z133).

„single page redirect not a biography article“: Die Einzeltrefferseite wurde nicht als Volltextartikel erkannt. Dies ist ein Hinweis darauf, dass sie umgestaltet wurde (Z143f.).

„Odd frame-set as response“: Die Antwort der Query-URL ist kein Frameset bzw. das Frameset enthält nicht die erwarteten Komponenten (Z160ff.).

„Getting results-list raised provider server error“: Die Ergebnisliste konnte nicht geladen werden. (Z164ff.)

„Odd content of response“: Die Antwort konnte weder als Einzeltrefferseite noch als Ergebnisliste erkannt werden. Dies ist ein Hinweis auf ein komplettes Relaunch der Trefferliste. (Z192)

„Can't find a results span“: Es wurde keine numerischen Informationen zur Ergebnismenge gefunden. (Z199)

„Couldn't find data to extract from results page“: Es wurden zwar numerische Daten über die Ergebnismenge, aber keine Metadaten erkannt. Sehr wahrscheinlich muss das `RecordPattern` aktualisiert werden. (Z203)

„Couldn't get next page“: Es konnte die nächste Seite der Ergebnisliste nicht geladen werden. (Z215f.)

„Getting next page raised a server error“: Die Antwort auf die Anfrage der nächsten Trefferseite ist ein Fehlercode. Dies weist auf einen Fehler in der URL hin. (Z224f.)

3.5.6 Testerfahrungen

Zuerst wurde die Suche nach „world“ getestet. Nachdem sie ca. 30 Sekunden benötigte, wurden 13036 Treffer anbieterseitig gefunden, 100 geholt. Wurde „peace“ als zweiter Term hinzugenommen, reduzierte sich diese Menge auf nur 44. Der Grund dieser geringen Anzahl ist, dass entweder „EXACT“ oder „+AND+“ als Suchmodus im Volltextfeld eingestellt werden kann, ohne dass Anführungsstriche beachtet werden. Hier ist die bessere Einstellung abzuwägen und gegebenfalls die Änderung in (Z95) vorzunehmen. Die Kombinierbarkeit von Phrase und UND-Verknüpfung ist fraglich.

Der Term „richard“, beschränkt auf `title`, ergab 47 Ergebnisse.

Die Einzeltrefferseite verursachte im Browser manuell der Term „world“ im „title“. Der Spezialist hat hiermit jedoch Probleme: Er wirft die Ausnahme „single page redirect not a biography article“ (s.o.). Es liegt vermutlich in dem regulären Ausdruck begründet, weshalb der Artikel nicht als solcher erkannt wird. Es konnte nicht mehr berichtigt werden.

Die kombinierte Suche nach „world“ im Volltext und „miller“ im `title` lieferte 16 Treffer.

Schließlich wurde die Phrasensuche mit „warmest thanks“ im Volltext getestet. Hierbei wurden nur zwei Treffer gefunden.

3.6 Oxford Reference Online Premium (oxfordref)

3.6.1 Nutzungsbedingungen

Gleicher Wortlaut wie ANB-Bedingungen.[8]

3.6.2 Konfiguration

Dies ist der Inhalt der Konfigurationsdatei `configs/oxfordref.src`:

```
name = Oxford Reference
srctype = OxfordRef
base = http://www.oxfordreference.com/
```

[8] http://www.oxfordreference.com/pages/privacypolicy_and_legalnotices – 12.06.2007 21:35

```
categories = 1 16 2 3 4 5 6 28 7 8 9 11 12 13 25 14 15 17 24 18 \
             19 20 21 22 23

<RecordTemplate>

<variantree>

RECROOT <<DEF
<div class="bodyentry">
<a href="##ENTRY_URL##">
##LEMMA##
</a></div>
<div class="main">##DESCRIPTION##</div>
<div class="global">
<i><a>##SOURCE##</a></i>
<a>##SECTION##</a>
</div>
DEF

</variantree>

<regextract />

<distribute />

</RecordTemplate>
```

3.6.3 Ablauf

OxfordRef kennt Sachkategorien, in die alle Artikel eingeordnet sind. Der Benutzer kann auf der Verlagsoberfläche seine Suche auf bestimmte Sachkategorien einschränken. Für die automatisierte Suche jedoch werden alle Kategorien ausgewählt, was in der Konfigurationsdatei geändert werden kann.[9] Die ID-Nummern der Sachkategorien wurden aus dem Quelltext der erweiterten Suchmaske ermittelt. (Z69)

Auch dieser Spezialist verwendet WWW::Mechanize. Es wird ein Objekt dieses Moduls mit aktiviertem `autocheck` instantiiert und auf die Identifikation als Mozilla-Browser eingestellt. (Z80-82)

Es werden nur folgende Default-Parameter gesetzt: Die Anzahl der Treffer pro Seite, `hits`, mit 100, sowie der Parameter `scope` mit „global“. (Z88ff.)

Die erweiterte Suche funktioniert mit der GET-Methode und stellt lediglich ein Suchfeld, ein Auswahlfeld über die Suchvariante (Full_Text, headings, person, date),

[9] Es ist möglich, mehrere OxfordRef-Quellen zu erzeugen, indem mehrere Konfigurationsdateien, die sich lediglich in der Kategorienzusammenstellung unterscheiden, erstellt werden.

sowie ankreuzbare Sektionen zur Verfügung, außerdem eine single-choice-Auswahl für den Suchtyp (concept, boolean, pattern) (vgl. [Lud07, S. 35ff.]).

Für die Suchvariante kamen nur `Full_Text` und `headings` in Betracht. Wird im Metalib-Query kombiniert nach beidem gesucht, was die Suchoberfläche nicht unterstützt, muss eine Nachfilterung erfolgen. (Z96-111)

Die Variante `People` wird nicht verwendet, da sie sich nicht nur auf die Lemmata, sondern auf überall im Text vorkommende Personennamen bezieht. Das heißt, sie passt nur manchmal auf die Suchfrage „Welche Informationen gibt es *über die Person*?“, während eine einschränkende Suche über mehrere Felder nicht möglich ist.

Die Variante *Dates* bleibt unberücksichtigt, da von Metalib keine Suche nach Daten unterstützt wird. Der Zweck der beiden Parameter `go.x` und `go.y`, welche ebenfalls in der auf eine manuelle Anfrage vom Anbieter zurückgeschickte GET-Adresse erscheinen, konnte nicht bestimmt werden. Sie scheinen jedoch ebenso abkömmlich zu sein wie die dynamisch zugewiesenen Parameter `ssid` und `time`.

Nachdem die URL abgeschickt wurde, wird sie im aktuellen Zustand des Spezialisten gespeichert. (Z120-122)

Nach der Überprüfung des Antwortcodes auf Erfolg werden die numerischen Daten über die Treffermenge aus dem Code extrahiert und ebenfalls gespeichert. Die Gesamttreffermenge kann aufgrund eines anbieterseitigen Limits 1000 nicht überschreiten. Während auf der Website die Überzahl mit „1000+“ angezeigt wird, kann dieses „+“ nicht mit übernommen werden, da der extrahierte Wert numerisch sein muss, damit Vergleiche funktionieren. (Z138-149)

Nach der Datenextraktion durch das untergeordnete HTML::StreamFisher-Objekt wird nach einem Link auf der Trefferseite, der den Text „Next“ enthält. Dieser wird verfolgt und einer neuen Iteration unterzogen, wenn weder das Limit noch die Gesamtanzahl der Treffer erreicht wurde. (Z166-197)

Im Falle einer kombinierten Suche werden die extrahierten Sätzen nun einer Nachfilterung unterzogen. Nachdem die ursprüngliche Anzahl numerisch gespeichert wurde, wird jeder Satz einzeln aus der internen Ablage von HTML::StreamFisher geholt (Z210) und das Lemma auf das Vorkommen des/der auf `title` qualifizierten Suchterms(-e) hin untersucht. Nur wenn dieser Check positiv ausfällt, wird er nach Dublin-Core umgeformt und in der Datensatz-Liste gespeichert. (Z214-218)

Abschließend wird die Referenz auf die Liste der Datensätze zurückgegeben.

3.6.4 Feldbelegungen

Das Dublin-Core-Element `title` wird mit dem `LEMMA` belegt (Z239). `description` wird zusammengestellt aus den drei Positionen `DESCRIPTION`, `SOURCE` und `SECTION` (Z241ff.).

Die Links zu den Artikeln sind mit der Session-ID verknüpft. Um nicht zu riskieren, dass die Gültigkeit des Links verfällt, wird der in einen Direktlink umgeformt, wie er in der Artikelansicht im abschließenden Zitiervorschlag angeboten wird, und der sich nur durch den Wegfall der Parameter `srn` und `ssid` und der Ankerreferenz `#FIRSTHIT` vom Link auf der Trefferseite unterscheidet.

3.6.5 Diagnostik

„Search raised a server error“: Die Antwort auf die Anfrage der Ergebnisseite ist ein HTTP-Fehlercode. Hinweis auf eine fehlerhafte Query-URL, wenn es sich um die erste Trefferseite handelt. (Z130)

„Odd content of response“: Die Antwort wurde nicht als Trefferseite erkannt. (Z157)

„Can't find any data to extract“: Von der Ergebnisseite konnten keine Metadaten extrahiert werden. Hinweis auf eine Umgestaltung der Ergebnisseite. (Z159f.)

„Can't find the next-page link“: Obwohl noch nicht die Gesamtanzahl der Ergebnisse erreicht wurde, ist kein Link auf die nächste Trefferseite zu finden. (Z168ff.)

„Couldn't get next page“: Nächste Trefferseite konnte nicht geladen werden. (Z176f.)

3.6.6 Testerfahrungen

Die Suche nach „christian“ im Volltext ergab 1000 Treffer (oder mehr, s.o.). Verbunden mit „atheism“ reduzierte sie sich auf 48 Treffer. „atheism“ als `title` lieferte nur 30. Diese wiederum verbunden mit „agnosticism“ im Volltext ergab nur 9 Ergebnisse. Die Phrasensuche funktionierte jedoch nicht erwartungsgemäß. Gesucht wurde hierbei nach ärguments are usually directed“. Geliefert wurden 64 Ergebnisse, die der Treffermenge der Suche mit getrennten einfachen, UND-verknüpften Suchtermen entsprechen. Dieses Problem wurde korrigiert.

3.7 Oxford Scholarship Online (oxfordsch)

3.7.1 Nutzungsbedingungen

Gleicher Wortlaut wie ANB-Bedingungen.[10]

3.7.2 Konfiguration

Folgendes ist der Inhalt der Konfigurationsdatei `configs/oxfordsch.src`:

```
# Name of the Resource
name = Oxford Scholarship Online

# Name of the Specialist subclass
srctype = OxfordSchlr

# base URL
base = http://www.oxfordscholarship.com/
```

[10] http://www.oxfordscholarship.com/oso/public/privacy_policy.html – 12.06.2006

```
# POST-Method form file
postformext = search/advancedquery

# Name of the form in the file
postformname = search

<RecordTemplate>

<variantree>

RECROOT <<DEF
    <div class="keylist-search">
    <img />
    <a href="##TOC_URL##">##TITLE##</a>##TITLE?##</div>
    <div class="list-search">##AUTHOR##</div>
    <div class="list-search">##SECTIONYEAR##</div>
    ##[ACCESS_FULLTEXT?]##
    </div>
    ##[CHAPTER?][CHAPTERPAGE?]## /* BOOKPAGE entfernt.
        Zwar weist ein Kommentar im Quelltext auf eventuelles Vor-
        kommen hin. Es wurde aber nie tatsächlich eines entdeckt.
        Außerdem hat HTML::StreamFisher (v0.0.42) Probleme,
        BOOKPAGE und CHAPTERPAGE zu unterscheiden! */
DEF

CHAPTER <<DEF
    <div><div class="chapter-lede-title-search">
    ##CHAPTER_TITLE##
    </div><div class="search-options">
    <a href="##CH_ABSTRACT_URL##"></a>
    ##[ACCESS_FULLTEXT_CHAPTER?]##
    </div></div>
DEF

CHAPTERPAGE <<DEF
    <div class="search-options-page">
    <script>##CHAPTER_PAGE_URL##</script>
    </div>
DEF

BOOKPAGE <<DEF
    <div class="search-options-page">
    <script>##BOOK_PAGE_URL##</script>
    </div>
```

```
DEF

ACCESS_FULLTEXT <<DEF
    <div class="search-options">
    <a></a>
    <a href="##FULLTEXT_URL##"></a>
    </div>
DEF

ACCESS_FULLTEXT_CHAPTER <<DEF
    <a href="##CH_FULLTEXT_URL##"></a>
DEF

</variantree>

<regextract>
SECTIONYEAR Subject:\s(.+?),\s(\d{4})
CHAPTER_PAGE_URL "(\S+?)","(\d+)"\)\)
BOOK_PAGE_URL "(\S+?)","(\d+)"\)\)
</regextract>

<distribute>
SECTIONYEAR         = SECTION,YEAR
CHAPTER_PAGE_URL    = CHAPTER_PAGE_URL,CHAPTER_PAGE_NUM
BOOK_PAGE_URL       = BOOK_PAGE_URL,BOOK_PAGE_NUM
</distribute>

</RecordTemplate>
```

3.7.3 Ablauf

Wie die bisherigen Spezialist instantiiert auch OxfordSchlr zuerst ein WWW::Mechanize-Objekt mit aktiviertem `autocheck` und der Identifikation als Mozilla-Browser.

Die Homepage von Oxford Scholarship Online muss zum Anfang dediziert aufgerufen werden. Offenbar ist dieser Schritt notwendig, damit der das System authentisiert wird und die Volltext-Links in den Trefferlisten erscheinen. (Z81)

Die Suchmaske von Oxford Scholarship Online schickt die Daten nicht nach der GET-, sondern nach der POST-Methode. Dies heißt in der Konsequenz, dass die Suchmaske aufgerufen werden muss, um dann die Daten mittels der von WWW::Mechanize dazu bereitgestellten Methodenaufrufen in die richtigen Felder des Formulars einzutragen. (Z83-111)

Das so ausgefüllte Formular wird abgeschickt. Durch das POST-Verfahren ist es nicht möglich, eine URL für einen eventuellen späteren Zugriff seitens des Nutzers zu speichern, die die Query-Daten bereits enthält. Es kann daher nur die Basis-URL im

aktuellen Zustand gespeichert werden. (Z119ff.)

Weil der Link auf die nächste Trefferseite frühzeitig benötigt wird, um die seitens des Providers als geliefert deklarierte Ergebniszahl zu berechnen (hierfür gibt es keine explizite numerische Information im Quelltext), wird er bereits jetzt im Quelltext gesucht. (Z134ff.)

Die Angabe der Trefferzahl muss aus separaten Zahlen für unterschiedliche Trefferarten addiert werden. Der Spezialist verlässt sich hierbei nicht auf die genauen Benennungen dieser Trefferarten (books, chapters, pages), auch nicht auf das Vorhandensein des Plural-*s*. Er lokalisiert stattdessen den kleinstmöglichen Bereich im Quelltext (Z146), in dem diese Daten erscheinen und extrahiert aus ihm alle Zahlen, auf denen ein beliebiges Wort und dann „found“ folgt. (Z144-161)

Nach der Extraktion der Daten über das Retrieval-Ergebnis werden vom dem Spezialisten angeschlossenen HTML::StreamFisher-Objekt die Metadatensätze extrahiert (Z172f.).

Ist weder das eingestellte Limit zu extrahierender Datensätze, noch die Gesamtanzahl der Treffer erreicht, wird die nächste Ergebnisseite geladen und einer erneuten Iteration unterzogen. (Z175-201)

Abschließend wird eine Referenz auf die Liste der ins Dublin-Core-Format überführten Metadatensätze zurückgegeben.

3.7.4 Feldbelegungen

Das Element `title` wird aus der Position `TITLE` gespeist. Ist außerdem ein Kapiteltitel vorhanden, wird es erweitert um den Trenner „ / “ und die entsprechende `CHAPTER_TITLE`-Position. (Z218-221)

Die Position `AUTHOR` kann mehrere Namen enthalten. Daher wird sie am Semikolon aufgesplittet und die einzelnen Vorkommen in getrennte `creator`-Elemente geschrieben. (Z224)

Im Element `description` erscheint lediglich `SECTION`. Der Text in der Position `YEAR` kommt in das Element `date`, für das das encoding scheme `YYYY` angegeben wird. (Z227-233)

Das Element `identifier.uri` wird aus einer von drei möglichen Positionen gefüllt: `CH_FULLTEXT_URL`, `CHAPTER_PAGE_URL` und `FULLTEXT_URL`. In allen Fällen wird die relative URL in die absolute Form überführt. (Z236-241)

Als `format` wird „text/html“ angegeben.

Ein Digital Object Identifier (DOI) ist wie bei den anderen Spezialisten kein Bestandteil eines Trefferdatensatzes. Dafür könnte die ISBN-13 aus der URL extrahiert werden, wenngleich ohne Trennstriche. Das dafür am besten geeignete Element ist `identifier` verbunden mit dem encoding scheme „ISBN“, bzw. das Feld `identifier.isbn`. Entsprechende Änderungen sollten in der Methode `record2dc()` vorgenommen werden.

3.7.5 Diagnostik

„Search raised a server error“: Die Antwort auf die Anfrage einer Ergebnisseite war nicht erfolgreich (HTTP-Fehlercode). Dies ist ein Hinweis auf fehlerhafte Formulardaten. (Z129f.)

„Odd content of response“: Die Antwort wurde nicht als Ergebnisseite erkannt. Vermutlich stimmt der reguläre Ausdruck für die Daten zur Ergebnismenge nicht mehr. (Z170)

„Can't find any data to extract“: Es konnten keine Metadaten von der Ergebnisseite extrahiert werden. Das `RecordPattern` muss sehr wahrscheinlich überholt werden (Z172f.)

„Couldn't get next page“: Die nächste Trefferseite konnte nicht geladen werden. Die URL ist vermutlich defekt. (Z181f.)

3.7.6 Testerfahrungen

Eine Suche nach dem einfachen Term „world“ im Volltext ergab 500 Treffer (100 geholt, 1 Appendix). Dies gilt auch mit den UND-verknüpften Zusätzen „century“ und „nature“, die schrittweise hinzugefügt wurden. Erst ein vierter Suchterm, „computational“, konnte die Menge auf 54 reduzieren.

Eine Suche nach „analysis“ in `title` lieferte 96 Treffer. Unerwarteterweise vergrößert sich die Treffermenge auf 164 (100 geh.), wenn sie mit „computational“ im Volltext ergänzt wird. Letzteres allein liefert wiederum 500. Daher kann eine irrtümliche ODER-Verknüpfung ausgeschlossen werden.

Die Phrasensuche nach „computational analysis“ ergab 21 Treffer. Beide Terme lediglich mit UND verknüpft lieferte 500.

Die Zahl von 500 ist sehr wahrscheinlich eine anbieterseitige Beschränkung.

3.8 Springer E-books (springer)

SpringerLink bietet im verlagseigenen Webangebot lediglich eine Metadatensuche an. Die Suche in den Volltexten wurde indes an die Suchmaschine *Google* delegiert.

3.8.1 Nutzungsbedingungen

Die Terms of Use von *Google* bzw. der für das Projekt in Betracht gezogenen Schnittstelle *Google AJAX Search API* verbieten implizit die Einbindung von Google als Quelle in eine Metasuche:

> You agree that you will not, and you will not permit your users or other third parties to: (a) modify or replace the text, images, or other content of the Google Search Results, including by (i) changing the order in which the Google Search Results appear, (ii) intermixing Search Results from

> sources other than Google, or (iii) intermixing other content such that it appears to be part of the Google Search Results; or (b) modify, replace or otherwise disable the functioning of links to Google or third party websites provided in the Google Search Results.[11]

Da die Aushandlung einer Sondervereinbarung mit Google als wenig Erfolg versprechend beurteilt wurde, hat das Erwin-Schrödinger-Zentrum mit dem Springerverlag erfolgreich über die Indexierung der E-books durch das Fachinformationszentrum für Chemie Berlin verhandelt. Das FIZ Chemie hat sich bereit erklärt, der Universitätsbibliothek den Index und die entsprechende Suchtechnologie zur Verfügung zu stellen. (vgl. [Lud07, S. 98ff.])

3.8.2 Konfiguration

Für die Volltextsuche in den E-Books wird die bereits bestehende Suchtechnologie des FIZ Chemie „Guide“ verwendet, die sich zum Zeitpunkt der Fertigstellung der vorliegenden Arbeit noch in der Entwicklung befindet. Sie ermöglicht die Ausgabe der Ergebnisse in XML. Es wird dafür der Spezialist „Specialist::FizChemieGuide“ entwickelt, der nicht nur die Springer E-Books, sondern in Zukunft ggf. auch andere Verlagskollektionen ansprechen können soll, die vom FIZ-Chemie über Guides zur Verfügung gestellt werden. Diese Verlagskollektionen müssen sich in ihrer Konfiguration nur durch ihren Parameter `guide` unterscheiden.

```
name = SpringerLink eBooks
srctype = FizChemieGuide
base = "http://www.fiz-chemie.de/"
guide = ebooksguide
results-limit = 3 # Aufgrund von Performanzgründen eingeschränkt
```

3.8.3 Ablauf

Zu Beginn wird ein WWW::Mechanize-Objekt mit der Option `autocheck` erstellt. Es soll sich auch hier als ein Mozilla-Browser ausgeben, auch wenn dies aufgrund der Ausgabe der Ergebnisse als XML nicht notwendig ist.

Die Parameterliste für die HTTP-GET-Methode wird initialisiert mit `sn=query_submit_xml` (`query_submit` für HTML-Antwort) sowie `step` mit dem Wert, der in der `src`-Datei für `results-limit` definiert wurde (Z76ff.). Es folgt die Zusammenstellung des providerspezifischen Querys. Die einzelnen Terme werden in Anführungsstriche gesetzt und mit dem Wort `AND` verbunden (Z84ff.). Die URL wird zusammengestellt und abgeschickt. (Z91ff.)

Diese Art der Query-Übergabe soll in Zukunft geändert werden. Das kommende Verfahren wird bis auf die fehlende Angabe des Operators dem gleichen, das NetLibrary anwendet: Die Feldangabe eines Suchterms wird indes mit `metaN`, die Wertangabe mit

[11] http://code.google.com/apis/ajaxsearch/terms.html – 12.06.2007 13:30

`fieldN` eingeleitet. N kann hierbei 1, 2 oder 3 betragen. „Die Einfeldsuche wird es erst in einer spaeteren Version geben“. [Sch07]

Nach Erhalt der Antwort wird aus dem Element mit dem XPATH-Pfad `//guides/data/hitlist/info/hits` die Anzahl der Treffer gelesen (Z131f.). Die URL wird kopiert, und die Kopie in eine URL für die entsprechende HTML-Seite umgewandelt, die der Nutzer aus dem Appendix aufrufen kann (Z138). Es werden sodann alle Elemente mit dem XPATH `//guides/data/hitlist/hits/hit` der Reihe nach an die Methode `record2dc()` übergeben.

Nun wird die `_update()`-Methode aufgerufen, wobei die Eigenschaften `results_total`, `results_got` und `first_results_page` gesetzt werden, letztere mit der obig generierten URI der für die Bildschirmausgabe geeigneteren HTML-Ausgabe.

Zum Schluss wird die Referenz auf die Liste der Datensatzstrukturen an die `retrieve()`-Methode der Basisklasse zurückgegeben.

3.8.4 Feldbelegungen

Aufgrund der wenigen Daten, die ein Datensatz gegenwärtig enthält, können nur die Dublin-Core-Felder `title` und `description` belegt werden. Hierbei müssen auch überflüssige Zeilenumbrüche und Leerzeichen am Anfang und am Ende der Zeichenfolgen entfernt werden, da sie sonst nicht vom verwendeten external hook für SRU erkannt werden. Elementattribute werden keine zurückgegeben. Eine URL zum Volltext ist gegenwärtig noch nicht in den Quelldaten enthalten.

3.8.5 Diagnostik

„Provider: 500 ...“ oder andere HTTP-Fehlercodes: Die Anfrage wurde nicht erfolgreich beantwortet. (Z122)

„Can't find number of records“: Die Gesamtanzahl der Trefferanzahl konnte nicht an der erwarteten Stelle des XML-Dokuments gefunden werden. (Z134)

3.8.6 Testerfahrungen

Große Probleme gibt es mit der Geschwindigkeit der Suche. Nur mit der Defaulteinstellung von drei Treffern pro XML-Dokument ist sie so hoch, dass die Suche innerhalb des von Metalib beschränkten Zeitfensters beendet wird. Aufgrund dessen wurde davon abgesehen, die Extraktion auf Folgedokumente auszuweiten, wofür der Provider den Parameter `start`[12] bereitstellt. Sollte dieses Problem beim Provider bestehen und dort gelöst werden, kann der für die Folgeseiten erforderliche Code aus einem der anderen Spezialisten übernommen werden.

Der erste Test war eine Suchanfrage mit einem unqualifizierten Suchterm: „world“. Die Antwort war nach 17 Sekunden ein SRU-Dokument mit 4 records. Diese Zahl ist erwartungsgemäß. Der vierte Datensatz ist der Appendix, der den Tester über eine

[12] Wertebereich: 1 - 1000

Treffergesamtzahl von 87 unterrichtet. Wurde ein zweites Wort „calculus“ hinzugenommen, war die Treffergesamtmenge lt. Appendix nur noch 7. Die KWIC[13]-Fragmente entsprachen der Suchanfrage. Die Phrasensuche funktioniert ebenfalls. Hierbei ergab die Suche nach „little difference conceptually“ einen Treffer. Eine Suche mit qualifizierten Termen ist noch nicht möglich, da seitens des Guides noch nicht unterstützt.

3.9 Specialist::Inactive

Dies ist ein Dummy-Spezialist, der lediglich auf die Frage `$s->is_available` *unwahr* antwortet und als Nebeneffekt gleich eine SRU-Diagnostic 2 wirft. Er besitzt keine Methode `_retrieve()`. Er wird durch die Konfiguration mit allen Quellen verknüpft, deren Konfigurationsdatei auf `.skip` enden.

Der Nutzen dieser Möglichkeit steht jedoch in Frage, da in Metalib selbst Quellen zeitweilig deaktiviert werden können.

3.10 Erstellung weiterer Spezialisten

Folgende Ausführung kann nur einen groben Leitfaden darstellen, da die Oberflächen von Anbietern elektronischer Texte , bzw. die dahinterliegenden Systeme höchst heterogen sind.

Zur Entwicklung neuer Spezialisten kann die Datei `Specialist/SKELETON` verwendet werden. Sie ist hierfür zu kopieren und unter passendem Namen mit der anbieterspezifischen Funktionalität zu versehen.

Im Verzeichnis `configs/` ist eine `.src`-Datei anzulegen, die den vollständigen Namen der Ressource enthält (Attribut `name`), sowie den Namen des verwendeten Spezialisten (`srctype`), wobei der Teil `Specialist::` weggelassen wird. Außerdem anzugeben ist die Basis-URL des Anbieters (`base`). Es liegt desweiteren beim Programmierer sorgfältig abzuwägen, welche Eigenschaften des Anbieters im Spezialist-Modul hart kodiert, und welche veränderbar in der Konfigurationsdatei hinterlegt und in einem Konfigurationslauf eingelesen werden.

Der Implementierung voraus geht eine eingehende Analyse der Suchoberfläche des Anbieters. Zuallererst sind jedoch die Nutzungsbedingungen des Anbieters daraufhin zu untersuchen, ob es notwendig ist eine Sondererlaubnis einzuholen.

Es ist zu ermitteln, ob und ggf. welche Vorkehrungen nötig sind, sich beim Provider zu authentisieren. Beispielsweise bei OxfordDNB musste hierfür eine spezielle URL aufgerufen werden. Für die Analyse der Abfragefunktionalität sollte die erweiterte Suche unbedingt der einfachen Suche vorgezogen werden, um alle Funktionen nutzen zu können. Die entsprechende Funktion des Browsers zeigt den Quelltext der Suchseite an. Das Starttag des Formulars ist `<form ...>`, das mindestens folgende Attribute hat: `method` gibt die HTTP-Methode der Abfrage, `GET` oder `POST` an, `action` die URL zum Serverprogramm des Anbieters, das für die Verarbeitung des Querys verantwortlich ist.

[13] keyword in context

Merkmal	**netlibrary**	**oxfordanb**	**oxforddnb**	**oxfordref**	**oxfordsch**	**springer**
Quelltyp (Klasse)	NetLibrary	OxfordANB	OxfordDNB	OxfordRef	OxfordSchlr	FizChemieGuide
rechtl. Abdeckung	Vereinbarung	Freiheiten innerhalb Institution eingeräumt				Vereinbarung
Query-Methode:	GET	GET	GET	GET	POST	GET
Query Volltext:	ja	ja	ja	ja	ja	ja
Query Titel:	ja	ja	ja	ja	ja	ja
Query Autor:	ja	nein	nein	nein	ja	nein
Query Verlag:	ja	nein	nein	nein	nein	ja
Query Identifier:	ja	nein	nein	nein	ja	nein
Query Schlagwort:	ja	nein	nein	nein	ja	nein
Ergebnisformat:	HTML	HTML	HTML	HTML	HTML	XML
Extrahiert Titel:	ja	ja	ja	ja	ja	ja
Extr. Autor:	ja	nein	nein	nein	ja	nein
Extr. Beschreibung:	ja	ja	ja	ja	nein	nein
Extr. Verlag:	ja	nein	nein	nein	nein	nein
Extr. Link zum Volltext:	ja	ja	ja	ja	ja	nein
Extr. DOI:	nein	nein	nein	nein	(ISBN)	?
Erkannte Sätze („world“):	80%	110%	100%	100%	100%	100%
Link auf nächste Seite:	ja	ja	ja	ja	ja	nein
Test 1 Wort Volltext:	okay	okay	okay	okay	okay	okay
Test mehr Wörter:	okay	okay	okay	okay	okay	okay
Test in „title“:	okay	okay	fast okay	okay	okay	n.m.
Test kombiniert:	okay	okay	okay	okay	okay	n.m.
Test Phrase:	zuviel	okay	Probleme	okay	okay	okay

Tabelle 3.1: Zusammenfassender Vergleich der Spezialisten

Das anschließende Vorgehen ist abhängig von der HTTP-Methode entweder die Ermittlung der Query-Parameter bei `GET`, um sie an die URL zum Verarbeitungsprogramm anzuhängen. Bei `POST` kann entweder die WWW::Mechanize-Methode `submit_form()` oder die LWP::UserAgent-Methode `post()` verwendet werden.

Für die Ermittlung und Eleminierung der *funktionalen* Query-Parameter bei `GET` kann eine manuelle Anfrage gestellt und schrittweise die URL in der Adresszeile auf die *semantischen* (d.h. die eigentliche Suchfrage repräsentierenden) Query-Parameter reduziert werden, solange die gleichen Ergebnisse geliefert werden. Der Rest sei das Aggregat aus notwendigen funktionalen Parametern und den semantischen Parametern. Erstere sollten nun als Default-Parameter definiert werden, bevor letztere mithilfe der Methoden des `IQuery`-Objekts bestimmt werden. Hierbei ist zu analysieren, welche Strategie der Provider zur internen Abbildung des Problems anwendet: Bei der Mehrheit der implementierten Spezialisten erfolgt die UND-Verkettung mehrerer Suchterme innerhalb eines oder verschiedener Felder (s. z.B. OxfordSchlr). Andere verwenden einen mehr analytischen Aufbau (s. NetLibrary). In jedem Fall sind die einzelnen Feldnamen aus dem Quelltext des Suchformulars zu ermitteln.

Die Methode `what_is_supported()` ist mit der unterstützten Funktionalität zu füllen, so dass diese sie an entsprechener Stelle im Programmablauf propagieren kann.

Nach Abschicken der Anfrage ist die URL via `_update( first_page_url => ...)` im aktuellen Zustand zu speichern.

Ebenfalls im aktuellen Zustand sind die numerischen Daten über die Ergebnismenge – d.h. die Position des letzten Treffers auf der Seite (`results_got`) sowie die Gesamttrefferseite (`results_total`) – zu speichern, die aus der Antwortseite zu extrahieren sind. Von diesen Werten hängt die korrekte Beurteilung der Treffermenge bzw. die Appendizierung ab.

Ist das Ausgabeformat HTML, erleichtert das der GatherYSe-Distribution beiliegende, zukünftige CPAN-Modul HTML::StreamFisher die Datenextraktion aus dem Quelltext. Um es zu verwenden, ist in der Konfiguration ein `RecordPattern`-Block mit den untergeordneten Blöcken `variantree`, `regextract` und `distribute` anzulegen. Zu beachten ist, dass bei der Patterndefinition das sog. „«HERE“-Quoting verwendet werden *muss*, da sonst die Rauten als Kommentarzeichen fehlinterpretiert werden:

```
RECROOT <<HERE
    /*
     * Patterndefinition.
     * Siehe ‘perldoc HTML::StreamFisher‘.
     */
    ...
    ##[SUBPATTERN?]##   /* funktioniert */
    ...
HERE

# Folgendes funktioniert nicht!
SUBPATTERN <th><a href="##URL##">##SERIES##</a></th>
```

```
# ....................^ Nur bis hier würde gelesen
```

Ist die Antwort ein Frameset, von welchem nur eine Komponente die Ergebnisseite darstellt, muss dies gesondert behandelt werden. Beispielhaft demonstriert wird dies in `Specialist::OxfordDNB`.

Die extrahierten Daten können im Speicher von HTML::StreamFisher belassen und erst später geholt werden, wenn alle Ergebnisseiten verarbeitet wurden.

Solange weder das eingestellte Limit an zu holenden Datensätzen, noch die Treffergesamtmenge erreicht wird, muss der Spezialist den Link auf die nächste Seite suchen, diesen aufrufen, den Zähler für `results_got` entsprechend erhöhen und die Metadaten extrahieren, bis eine jener Bedingungen erfüllt ist.

Konnten nicht alle Aspekte der gestellten Anfrage auf die anbieterspezifische Suchfunktionalität abgebildet werden, ist eine Nachfilterung erforderlich. Es wird empfohlen, die notwendigen Checks an den originalen, von Streamfisher gelieferten Daten durchzuführen.

Schließlich sind alle Datensätze nach Dublin-Core zu überführen. Es hat sich bewährt, diese Aufgabe in eine eigene Methode `record2dc()` auszulagern. Die Umformungen müssen sowohl den attributiven anbieterübergreifenden, als auch den mit der Methode `get_dc_attributes()` definierten anbieterspezifischen Vorgaben entsprechen. Wo dies nicht möglich ist, können datensatzspezifische Ausnahmen in einen Datensatz kodiert werden.

Als Rückgabewert wird die Referenz auf die Liste der umgeformten Datensätze erwartet, nicht die Liste als solche.

3.11 Test-Interface

Die GatherYSe-Distribution enthält eine einfache CQL[14]-basierte Suchmaske für Tests am GatherYSe-System und an den konfigurierten Spezialisten ohne eventuelle Störfaktoren, die vom Bibliotheksportal herrühren können. Das Test-Interface wird über die Datei `test.pl`[15] im Webbrowser aufgerufen. Als Antwort auf eine Anfrage wird das reine, unformatierte XML-Dokument von GatherYSe geliefert.

Bessere Möglichkeiten der Fehlersuche eröffnet folgender Befehl. `name` ist durch den Kurznamen des Spezialisten zu ersetzen, `query` durch die CQL-Anfrage.

```
GatherYSe$ perl debugSpecialist name "query"

Loading DB routines from perl5db.pl version 1.28
Editor support available.

Enter h or 'h h' for help, or 'man perldebug' for more help.
```

[14] http://www.loc.gov/standards/sru/cql/index.html – 21.07.2007
[15] http://.../cgi-bin/GatherYSe/test.pl

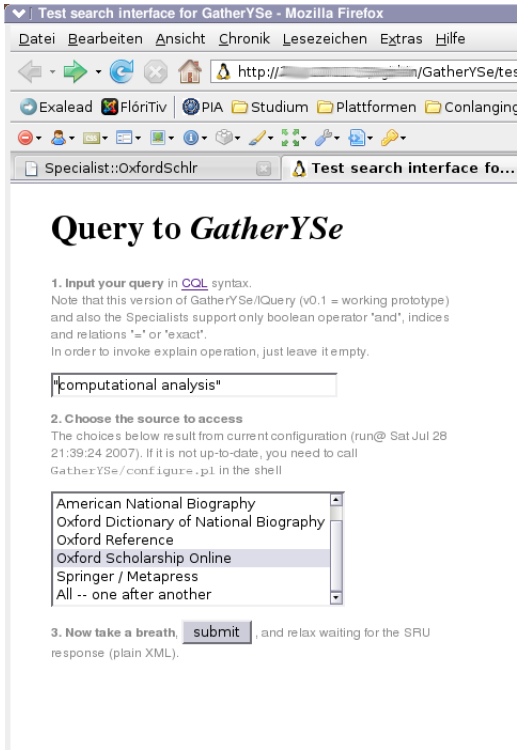

Abbildung 3.1: Suchinterface

```
Sending QUERY: query TO Long Specialist Name
main::(debugSpecialist:6):      my ($specialist,$query);
  DB<1>
```

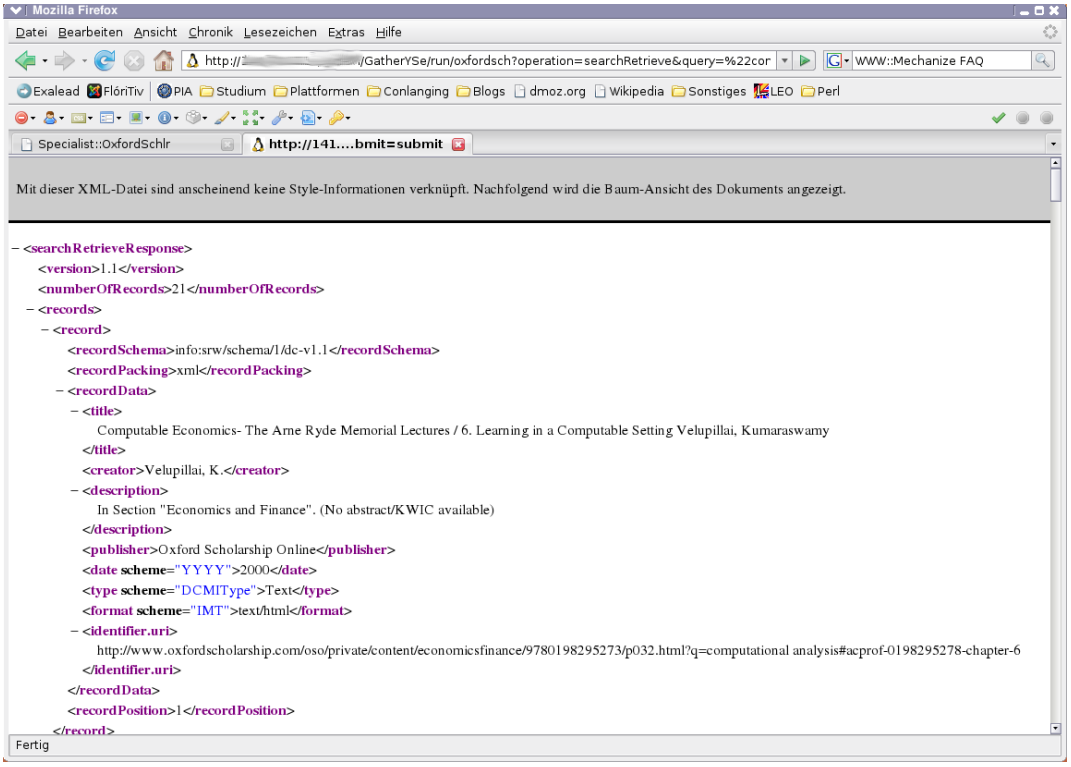

Abbildung 3.2: XML-Ausgabe nach dem SRU-Protokoll

4 Die Integration von GatherYSe in Metalib

Das Bibliotheksportal der Humboldt-Universität zu Berlin ist Exlibris Metalib. Es befähigt den Nutzer, über ein einheitliches Suchinterface zahlreiche Datenbanken und Ressourcen gleichzeitig abzufragen. Dabei kann er nach Abschicken der Anfrage mitverfolgen, welche Angebote in welcher Reihenfolge wie viele Ergebnisse zur gestellten Suchfrage zurückliefern. Die einzelnen Ergebnissets werden vereinigt und einheitlich von Metalib dargestellt. Eine zusätzliche nützliche Funktion stellt das SFX-Modul bereit. Dies ist ein Link Resolver im Sinne des OpenURL-Standards, der abhängig von den übergebenen Meta- und Authentifizierungsdaten alle für eine Einheit verfügbaren Services wie etwa Volltextzugriff oder Fernleihe anzeigt.

Die Resource wird mit dem „Universal Gateway“ angesprochen, ein Metalib-Programmmodul, das sich um die ressourcenspezifische Übersetzung der Suchanfrage sowie um die eintreffenden Datensätze kümmert, oder sie wird direkt angebunden. Grundsätzlich unterscheidet Metalib drei Ressourcenarten:[1]

- *MetaLib Link Resources* – einfache Verknüpfung zur Suchoberfläche des Providers
- *Metalib Search and View Resources* – Metalib kann eine Suchanfrage an die Ressource stellen und ihre Ergebnisse verarbeiten.
- *Metalib Search and Link Resources* – Metalib kann nur eine Suchanfrage an die Ressource stellen, während der Benutzer zur Ansicht der Treffer via einem entsprechenden Link auf die providereigene Seite gehen muss.

Es wird die Integration von GatherYSe als *Metalib Search and View Resources* angestrebt.

Die Suchergebnisse aller Ressourcen werden in einer temporären Datenbank mit dem Label `vir01` gespeichert. Ein jedem gespeicherten Datensatz zugewiesener Wert ist die SID, welche die Zugriffs-/Systemnummer der Originaldatenbank enthalten kann. Die Dokumentation empfiehlt die regelmäßige Löschung dieser Datenbank, welche auch zur Einhaltung der Nutzungsbedingungen der verschiedenen E-Book-Provider – die unter anderem die Datenspeicherung ihrer Inhalte verbieten – dringend durchzuführen ist.

[1] Ex-Libris: Metalib Resource Management Guide vom 16. Oktober 2005 für Metalib Version 3.13, Service Pack 69. – Nicht öffentlich verfügbar

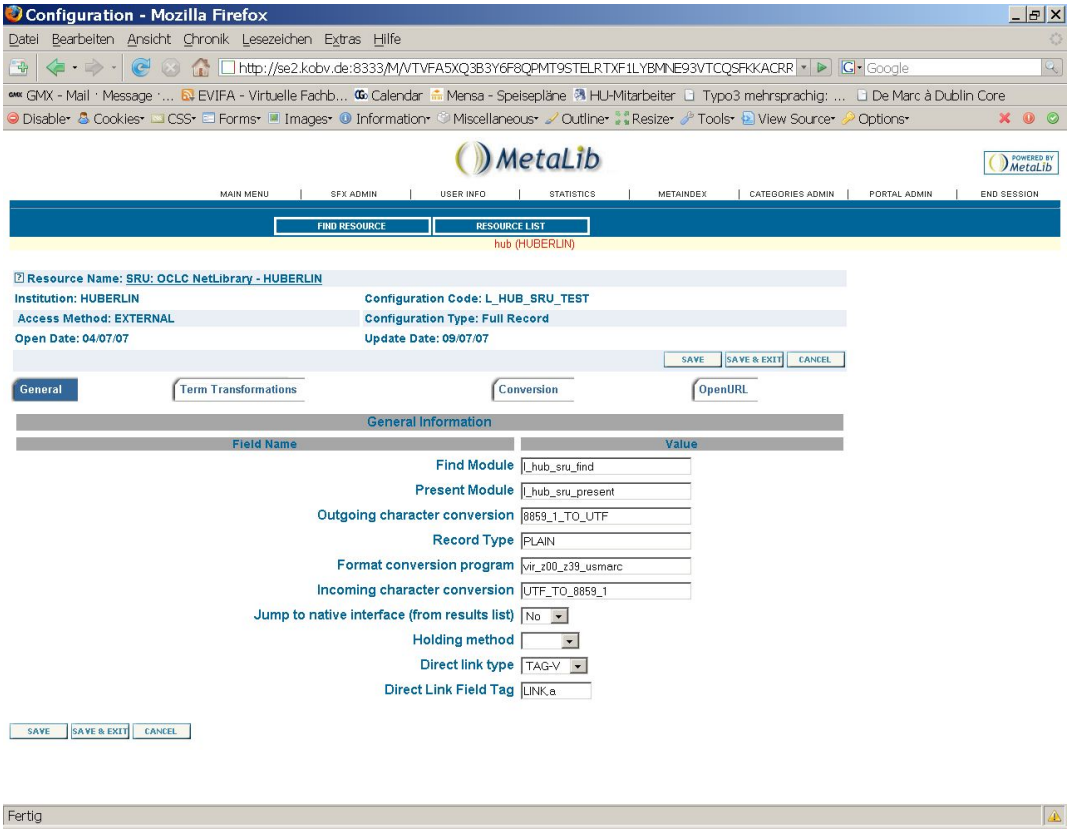

Abbildung 4.1: Konfiguration des external hooks[2]

Metalib der Version 3.13 enthält keinen direkten SRU-Support, erst die kommende Version 4 soll dieses Protokoll nativ beherrschen. Damit die GatherYSe-Quellen in Metalib integriert werden können, muss auf einen *external hook* zurückgegriffen werden, also einem externen Programm, dass „fremdartige“ Quellen in das von Metalib verstandene USMARC-Format übersetzt. Dieser external hook wurde von Goverdovski über Skripte der finnischen Nationalbibliothek realisiert, die sich – gegenwärtig in der Version 2.0 – noch in der Weiterentwicklung befinden. [Nat]

Das Metalib Administrationsinterface wird im Browser aufgerufen über die Adresse `http://<IP or hostname of MetaLib server:port>/M`. Es erscheint eine Abfrage nach User-ID und Passwort.

Von dort aus wird die IRD-Maske[3] aufgerufen. Hier kann der Administrator oder andere dazu autorisierte Personen Registrierungsdaten von Ressourcen ändern oder neue Ressourcen erstellen.

[2] Quelle dieser und aller weiteren Screenshots dieses Kapitels: Die Metalib-Testinstanz der Humboldt-Universität Berlin

[3] Information Resource Database Record

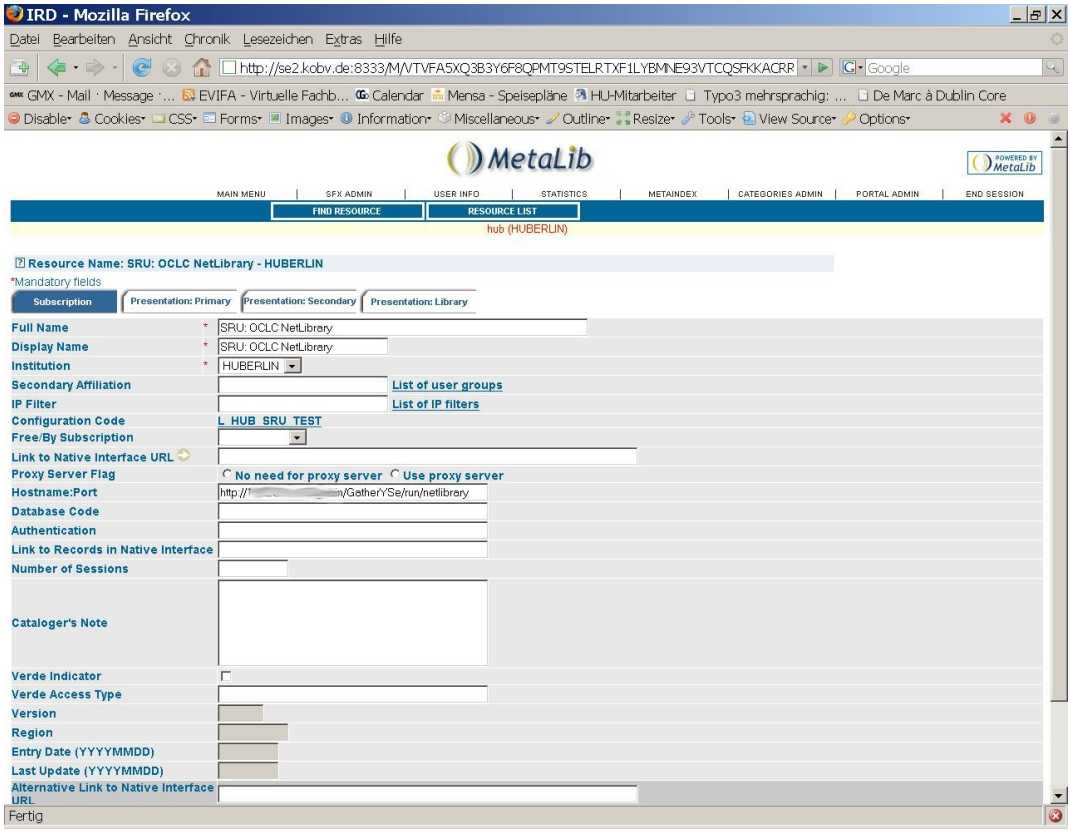

Abbildung 4.2: Subscription Tab

Es stehen vier Register zur Verfügung:

- Subscription Tab
- Presentation: Primary Tab
- Presentation: Secondary Tab
- Library Tab

Nur die ersten drei bedürfen einer näheren Betrachtung und müssen für die GatherYSe-Spezialisten ausgefüllt werden. Das *Library Tab* ist allein für die Einbindung von OPACs gedacht.

4.1 Subscription Tab

Full Name Einzutragen ist der vollständige Name der Ressource.

Display Name Der dem Nutzer angezeigte Name kann ein anderer sein.

Institution, Secondary Affiliation, IP Filter Dies wird benötigt für die Authentifizierung bei der Ressource. Da die E-Book-Anbieter selbst ebenfalls die IP-Authentifizierung verwenden, sollte diese Methode auch in Metalib aufgegriffen werden, damit durch die Verfügbarmachung der Ressourcen nach außerhalb nicht die Nutzungsbedingungen der Anbieter verletzt werden. Mit dem IP-Filter ist daher der Zugriff auf die Ressourcen auf den Campus der HU-Berlin einzuschränken.

Configuration Code Dieses Feld stellt die Verknüpfung zur Konfiguration der Ressource her.

Free/By Subscription Diese Option trifft nicht zu, da IP-Authentifizierung. Hier sind jedoch Potenziale bzgl. Athens und Schibboleth zu erkennen, indem hier die User ID und das Passwort für den Zugriff von außerhalb in das resultierende Formular eingegeben wird.

Link to Native Interface URL Verknüpfung zur Suchoberfläche des E-Book-Anbieters.

Proxy Server Flag Wird ein Proxyserver verwendet, muss seine Adresse hier eingetragen werden.

Hostname, Port Dieses Feld enthält die URL zum entsprechenden Symlink, der der GatherYSe-Quelle zugeordnet ist. Für Oxford Scholarship Online ist dies beispielsweise `http://{IP+Pfad}/GatherYSe/run/oxfordsch`.

Link to Records in Native Interface Dieses Feld bleibt leer, da die Records bei den Anbietern keine Ein-Datensatz-Ansicht haben.

Number of Sessions Diese ist im Rahmen der Nationallizenzen unendlich. Daher bleibt das Feld leer.

Verde Indicator Das Feld würde zur Anbindung an die ExLibris-Software Verde verwendet (elektronisches Record-Management System).

Verde Access Type Damit verbunden erfolgte die Angabe der verwendeten Schnittstelle wie HTTP, Z39.50, SRU und andere.

Alternative Link to Native Interface URL Dieses Feld könnte etwa eine Athens-spezifische URL enthalten.

Alternative Link to Records in Native Interface Dieses ebenfalls.

Rank Dieses Feld akzeptiert einen Wert zwischen 1-5, um in einer gemischten Trefferansicht die Datensätze verschiedener Ressourcen (E-Book-Anbieter) gegeneinander zu wichten.

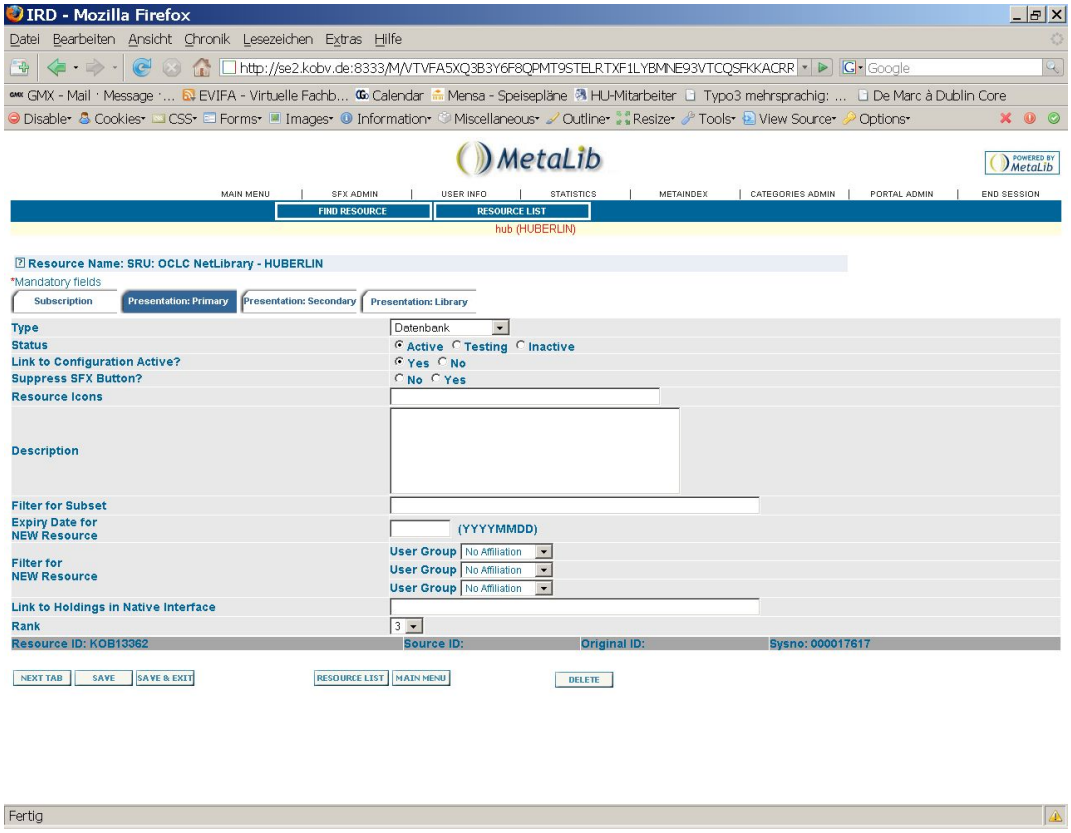

Abbildung 4.3: Primary Tab

4.2 Presentation: Primary Tab

Definition der allgemeinen Eigenschaften der Ressource, wie sie dem Benutzer angezeigt werden.

Resource Typ Eine passende Wahl ist hier „Volltext“.

Status Diese Auswahl bietet die Auswahl zwischen „Active“, „Testing“ und „Inactive“. Da für den Nutzer ggf. aussagekräftiger, sollte gegenüber der GatherYSe-Variante (s. 3.9) vorgezogen werden.

Suppress SFX Button Da bei den E-Book-Anbietern keine SFX-Funktionalität verwendet werden soll (keine Berücksichtigung von Inhalten, die nicht im Volltext lizenziert sind), kann diese Option auf „Y“ gestellt werden.

Resource Icons Hier können zusätzliche Symbole eingetragen werden, die neben dem (versteckten) SFX-Button erscheinen sollen. Zum Beispiel könnte hier im Rahmen von Partnervereinbarungen mit dem E-Book-Anbieter dessen Logo eingetragen werden.

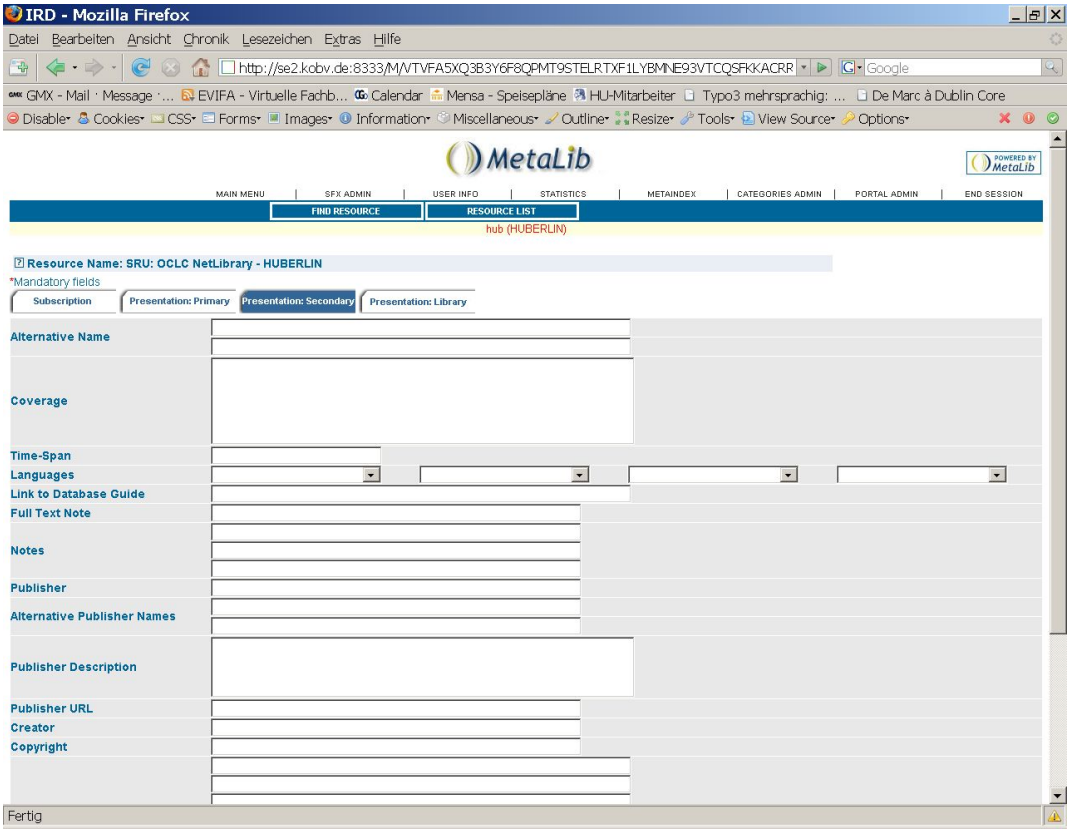

Abbildung 4.4: Secondary Tab

Description Dieses Feld nimmt eine Beschreibung oder Schlagwörter auf.

4.3 Presentation: Secondary Tab

Alternative Names Hier können andere mögliche Namen der Ressource eingetragen werden, nach denen ebenfalls über den Metalib-Datenbankfinder gesucht werden kann.

Coverage Dieses Feld kann belegt werden, wenn der Anbieter hierzu genaue Angaben macht.

Time-span Dieses Feld kann belegt werden, wenn der Anbieter hierzu genaue Angaben macht.

Link to Database Guide Die Spezifika der Suchfunktionalität des Anbieters kann nicht direkt vom Metalib-Nutzer gesteuert werden. Daher wird hier auch kein Link angegeben.

Full Text Note Dies wird für alle eingebundenen E-Book-Anbieter aktiviert.

Publisher Hierbei zu beachten ist die Produkt-zu-Hersteller-Abstraktion, z.B. „Oxford University Press“ für alle vier Oxford-Angebote.

Publisher URL Diese ist z.B. bei Oxford nicht identisch mit den URLs der E-Book-Angeboten (http://www.oup.com/)

Creator Dieses Feld ist nur zu besetzen, wenn ein dedizierter Ersteller des Inhalts der E-Book-Datenbank eines E-Book-Anbieters ermittelt werden konnte.

Copyright Notices Hier muss ein Hinweis auf die Nutzungsbedingungen des jeweiligen Anbieters erfolgen.

4.4 Eine Beispielsuche

Um eine Parallelsuche zu starten, muss auf der Metalib-Oberfläche über die obere Navigationsleiste die Seite *Suche in Datenbanken* aufgerufen werden. Im Menü *Ressourcenauswahl* ist „Suchen“ zu wählen und als Titel `SRU` einzugeben. Das Prefix „SRU:“ kann für die öffentlich freigeschaltete Anbindung entfernt werden, denn für den Nutzer ist es keine Information und kann ihn irritieren. Nachdem die sechs Ressourcen gefunden wurden, kann auf derselben Seite gleich die eigentliche Suchfrage abgeschickt werden.

4.5 Probleme

Es bestehen bei der Suche in Metalib in GatherYSe-Quellen zwei Probleme.

Es werden zum Einen nicht alle Felder aus der SRU-Antwort angezeigt. Aus 4.10 ist zum Beispiel ersichtlich, dass das Jahr und der Verlag bzw. (NetLibrary) fehlt, wobei letzteres aufgrund der Metalib-seitigen Quellenangabe weniger ungünstig ist. Um jedoch zumindest das Datum mit hineinzunehmen, muss ermittelt werden, in welchem Dublin-Core-Element der external hook das Datum tatsächlich erwartet. Es wird empfohlen, das ggf. erforderliche Remapping nicht im Spezialisten des Quelltyps, sondern in der Basisklasse vorzunehmen (Z199), so dass die Änderung an dieser Stelle leicht wieder rückgängig zu machen ist.

Zum Anderen besteht ein Problem mit der Geschwindigkeit. Alle external hooks bestehen bei Metalib aus zwei Komponenten:

- `sru*_find` ist verantwortlich für die Umwandlung der von Metalib erhaltenen Query-Daten in die CQL-Syntax und generiert die entsprechende URL gemäß SRU-Protokoll. Diese URL wird aufgerufen. Aus der Antwort wird jedoch allein die Trefferzahl extrahiert.

- `sru*_present` schickt dieselbe URL ein weiteres Mal, extrahiert nun aber alle enthaltenen Datensätze aus dem XML-Dokument und überträgt es in das von Metalib erwartete Format.

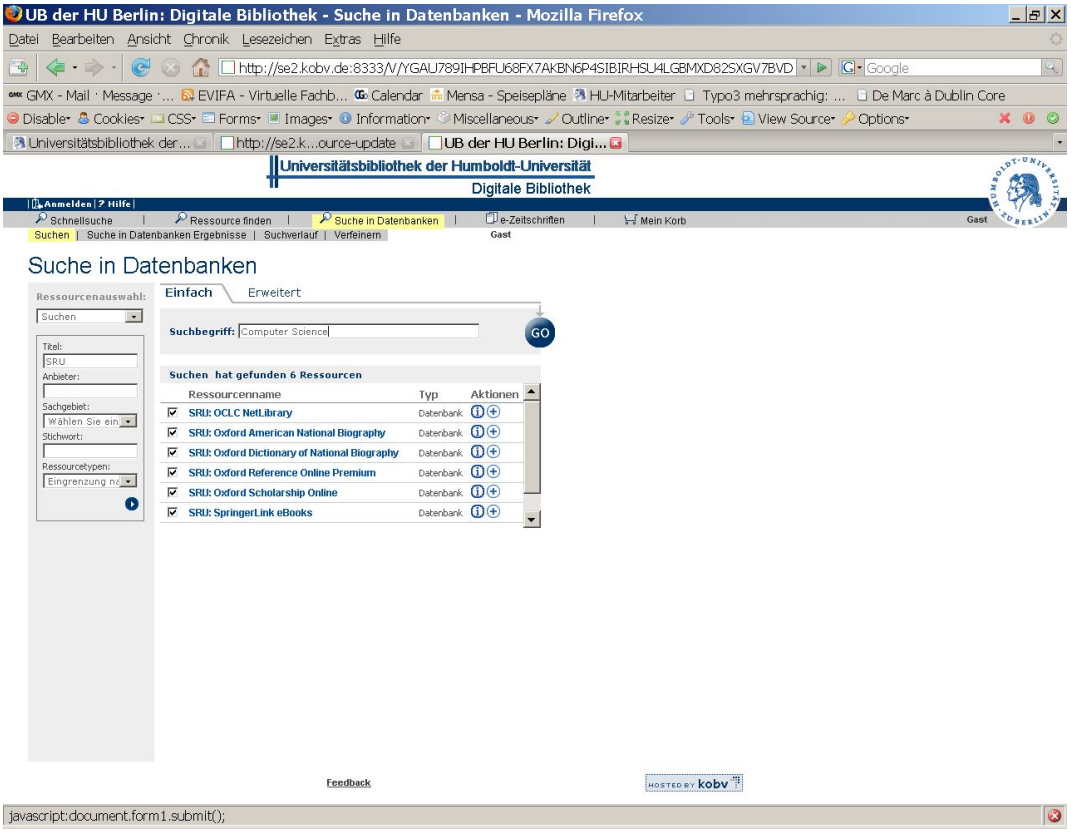

Abbildung 4.5: Ressourcenauswahl und Suchfrage

Die doppelte Suche ist problematisch, da GatherYSe prinzipbedingt um ein Vielfaches langsamer ist als eine Anfrage an eine Datenbank. Das Laden von Webseiten zum Zweck der Informationsextraktion ist im Vergleich zu einem Datenbankzugriff ineffektiv, da die eigentliche bedeutungstragende Information gegenüber der strukturellen und der Layout-Information mengenmäßig nur einen kleinen Teil ausmacht. Dennoch ist der zeitliche Aufwand der Extraktion aus Webseiten durch HTML::StreamFisher klein[4] gegenüber den Wartezeiten, bis eine Seite vollständig empfangen ist.

Das Metalib der Humboldt-Universität hat einen Timeout von sechzig Sekunden. Zu wenig, als dass GatherYSe-Instanzen ihre Arbeit rechtzeitig abschicken können. Eine Erhöhung dieses Timeouts geht wiederum zu Lasten der Benutzerfreundlichkeit des Metalib-Systems.

Abhilfe könnte ein Proxyserver[5] schaffen. GatherYSe vorgeschaltet könnte der Proxy die vom `find`-Skript erhaltene URL sowie die dazugehörige Antwort von GatherYSe

[4] Eine bis zwei Sekunden für eine Ergebnisseite der Größenordnung von 10 bis 100 Treffern

[5] Proxyserver speichern Internetseiten zwischen, um Wartezeiten zu verkürzen und den Traffic im Internet zu reduzieren

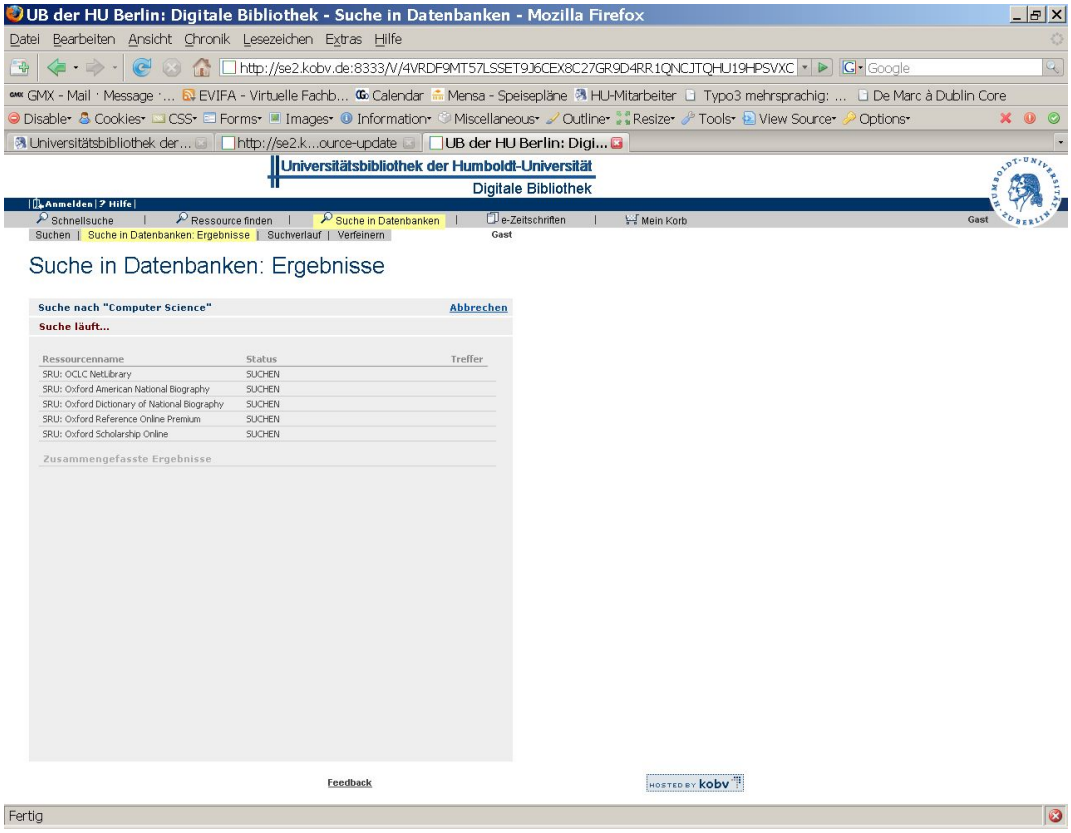

Abbildung 4.6: Fortschrittsanzeige 1 (Springer E-Books wg. zeitweiligen Problemen exkludiert)

zwischenspeichern, und letztere bei darauffolgender selbigen Anfrage vom `present`-Skript gleich an dieses zurück liefern. Dies hätte zur Folge, dass GatherYSe pro Nutzeranfrage und Quelle nur einmal gestartet wird und durch die resultierende Kapazitäteinsparung die Instanzen schneller ausgeführt werden.

Als provisorische Alternative zu einem Proxyserver – seine korrekte Konfiguration muss noch ermittelt werden – hat Lüttgau den external hook um einen Caching-Mechanismus erweitert. Hierbei speichert das Find-Skript die SRU-Antwort in einer Datei mit einem eindeutigen Namen und gibt diesen an Metalib zusammen mit den anderen Daten zurück. Da Metalib die Daten lediglich an das Present-Skript weiterleitet, kann dieses anhand der ID direkt auf die hinterlegte Datei zugreifen.

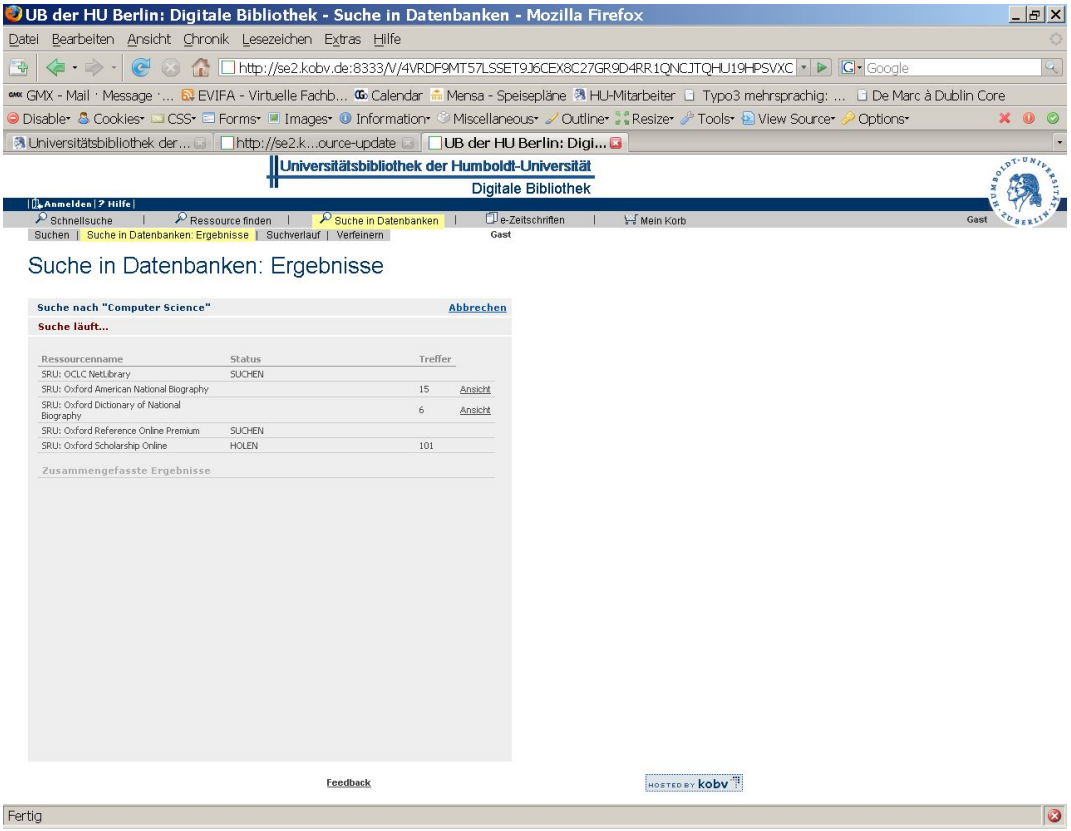

Abbildung 4.7: Fortschrittsanzeige 2

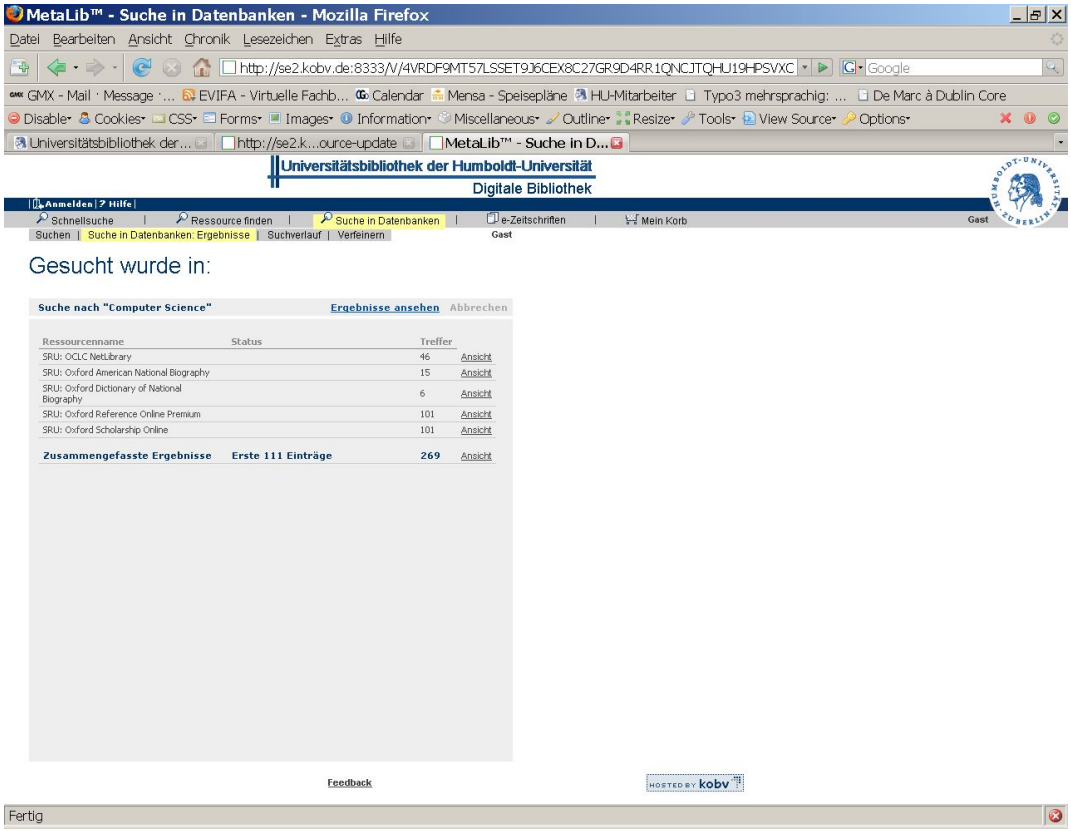

Abbildung 4.8: Finale Statusanzeige

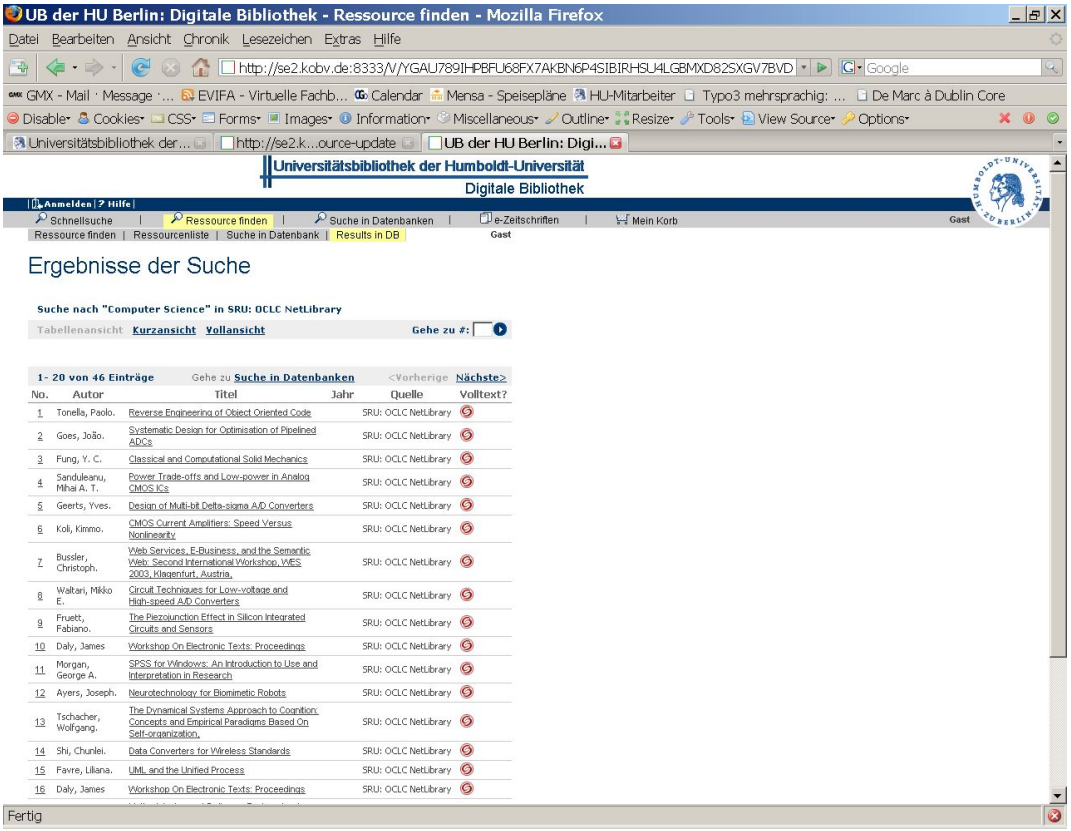

Abbildung 4.9: Ergebnisliste einer Quelle

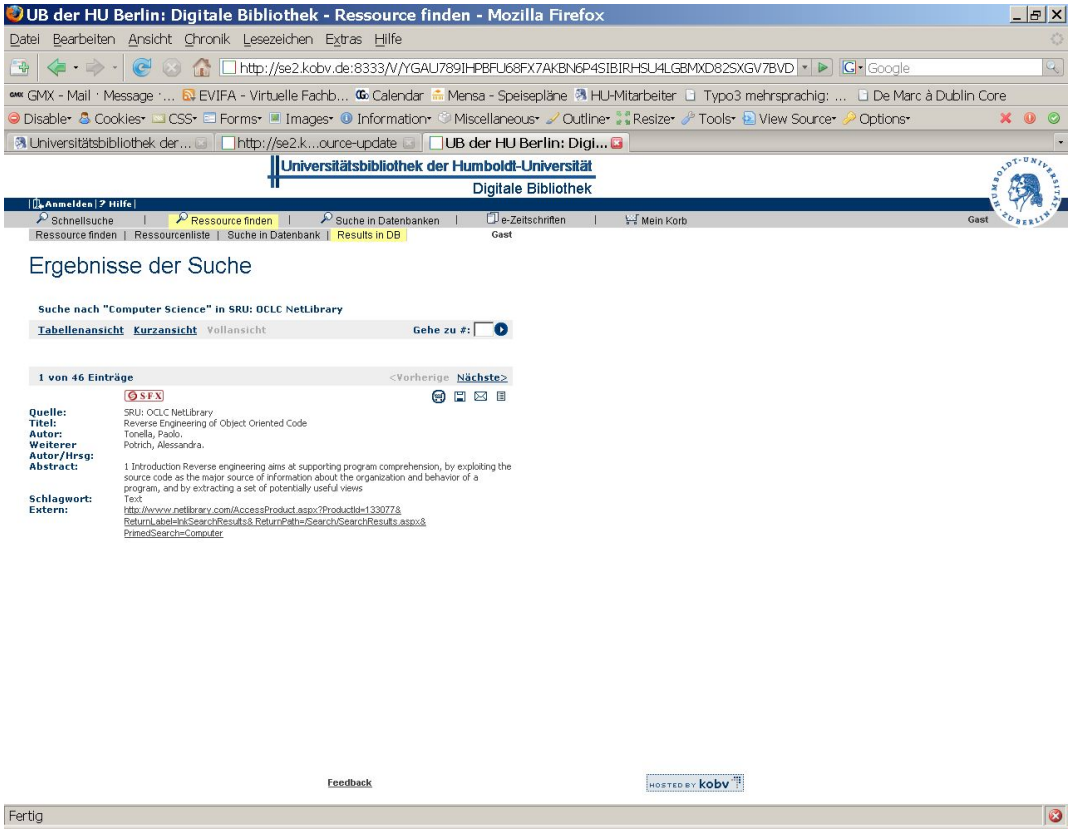

Abbildung 4.10: Detailansicht

5 Abschließende Betrachtung

5.1 Zusammenfassung

In der vorliegenden Arbeit wurde gezeigt, dass eine Anbindung von online über Anbieterwebsites verfügbaren E-Book-Kollektionen an Bibliothekssysteme wie Exlibris Metalib technisch möglich ist, um den Nutzern eine leistungsfähige Metasuche zur Verfügung zu stellen.

Die Universitätsbibliothek hat von verschiedenen E-Book-Anbietern Lizenzen erworben. Diese ermöglichen von sich aus den Zugriff auf ihre Kollektionen allein über eine webbasierte Suchoberfläche. Der typische Bibliotheksnutzer sucht jedoch meist nach Informationen, ohne sich auf ein bestimmtes Medium oder einen bestimmten Anbieter beschränken zu wollen. In der Konsequenz heißt dies, dass er auf der Suche nach Literatur auch die Suchoberflächen der einzelnen Anbieter der Reihe nach abrufen, kennen lernen und die Kollektion dort durchsuchen muss.

Diese Vorangehensweise ist unzumutbar, vor allem in Anbetracht der obersten Aufgabe einer Bibliothek, der Öffentlichkeit gesuchte Informationen *schnell*, *einfach* und *effizient* zur Verfügung zu stellen. Wenngleich die Zugriffsmöglichkeit im Rahmen einer Metasuche in einem deutlichen Konflikt mit dem Interesse an Autonomie seitens der Verlage steht, muss die Bibliothek auch in den zunehmend elektronisierten Informationswelten ihre Aufgabe wahrnehmen können, damit die Wertschöpfung der Gesellschaft aus dieser Einrichtung auch in Zukunft bestehen bleibt. Daher ist es zu begrüßen, dass einmal sie Eigeninitiative zeigt, den Nutzwert der für mehrere tausend Euro lizenzierten E-Book-Kollektionen zu optimieren und das Potenzial der – nicht neuen – Medienform zu entfalten. Ein wichtiger Aspekt dieses Potentials war die maschinelle Durchsuchung des Volltextes nach Stichwörtern und Phrasen, das bei Printmedien nicht möglich ist.

Im Rahmen der vorliegenden Arbeit sollte das Bibliotheksportal in diesem Sinne um die Fähigkeit erweitert werden, in den E-Book-Kollektionen parallel zu suchen und dem Nutzer die Ergebnisse in einem einheitlichen Format anzuzeigen. Priorität hatte hierbei die Suchmöglichkeit in den Volltexten, weshalb die Auswahl auf die volltextfähigen E-Book-Suchoberflächen beschränkt wurde.

Es wurde aus Gründen der Machbarkeit und der Nachhaltigkeit davon abgesehen, die Erweiterung – ihr offizieller Name: *GatherYSe* – direkt in das Metalib-System zu integrieren. Stattdessen fungiert GatherYSe seinerseits als ein gateway-artiger Server, der in Metalib als völlig normale Ressource eingebunden wird. Als Kommunikationsprotokoll zwischen Metalib und GatherYSe kommt SRU zum Einsatz. Zwar arbeiten viele Bibliothekssysteme immer noch mit Z39.50, doch handelt es sich hierbei um ein in die Jahre gekommenes, komplexeres Protokoll, das gegenwärtigen und vor allem

zukünftigen Herausforderungen nicht mehr wird bewältigen können. Aus diesen Gründen sowie aufgrund der begrenzten Zeit zur Implementierung bot sich der einfachere Z39.50-Nachfolger an, auch wenn die SRU-Unterstützung dem aktuell verwendeten Metalib-System als external hook hinzugefügt werden musste.

Es wurde versucht, GatherYSe durchgehend modular und objektorientiert zu strukturieren. Zu betonen ist hier sowohl die Trennung als auch das kontrollierte Zusammenwirken zwischen der Problemstellung als ein beliebig komplexes Gefüge von Anforderungen an die gesuchte Literatur (Suchfrage), und der Problemlösung als algorithmischer Prozess der bestmöglichen Abbildung dieses Gefüges auf die Funktionalität der Suchoberfläche. Hierfür ist das GatherYSe/IQuery-Modul zuständig. Ebenso wurde darauf geachtet, dass die einzelnen Spezialisten soweit wie möglich gemeinsames Verhalten aufweisen und die anbieterspezifischen Prozesse überschaubar an einem Ort konzentriert werden, ohne dass ihnen technische Grenzen und Beschränkungen durch die übergeordneten Komponenten auferlegt werden. Während das eigentliche Retrieval in einer Quelle von den einzelnen Spezialisten durchgeführt wird, liegt in der Verantwortung des Hauptprogramms nur noch, die internen Daten ins SRU/XML-Format zu konvertieren.

Außer Springer, bzw. der FIZ-Chemie-Guide, liefern alle Anbieter die Ergebnisse im instabilen, weil leicht veränderlichen HTML-Format. Es konnte in der kurzen Vorbereitungszeit kein frei verfügbares Modul gefunden werden, das den GatherYSe-spezifischen Anforderungen für die Metadatenextraktion aus dem HTML-Code genügt. Daher wurde die Eigenentwicklung namens HTML::StreamFisher unternommen, ein Webscraper auf Basis von leicht an anbieterseitigen Änderungen anpassbaren Patterndefinitionen.

Es konnte für alle ausgewählten Quellen grundsätzlich funktionierende Spezialisten implementiert werden. Besondere Herausforderungen stellten dabei unter anderem die Behandlung von Framesets (OxfordDNB), die analytische Query-Abbildung (NetLibrary) und die Nachfilterung von Suchergebnissen dar (OxfordRef). Alle Spezialisten wurden mehreren Tests unterschiedlicher Art und Komplexität unterzogen, welche sie weitgehend bestanden. An einigen beschriebenen Stellen sind Nachjustierungen notwendig. Die Guides des FIZ Chemie befinden sich gegenwärtig im Entwicklungsstadium, hier wird bald eine Revision des Spezialisten notwendig sein.

Es ist davon auszugehen, dass die Verlage Projekten wie GatherYSe, vor allem ihre tatsächliche Anwendung in der Bibliothek, nicht wohlgesonnen gegenübertreten werden, da sie ihre Autonomie gefährdet sehen. Doch auf jeden Fall verschafft diese Software den Bibliotheken eine gute Verhandlungsposition, wenn die Erlaubnis, GatherYSe zur Einbindung neuer E-Book-Kollektionen anzuwenden, ein Entscheidungskriterium zur Erwerbung einer Nutzungslizenz wird. Eine weitere Möglichkeit ist, dass GatherYSe als eine Art Druckmittel für Verlage wirkt, von sich aus eine SRU-Schnittstelle zu ihren E-Books anzubieten.

GatherYSe ist freie Software und steht unter der General Public Licence, Version 3 vom 29. Juni 2007, s. `COPYING`. Hiermit soll eine Weiterentwicklung von GatherYSe innerhalb einer Entwicklercommunity begünstigt werden. Auch für die nutzenden Einrichtungen ergeben sich hieraus große Vorteile [RVRK05].

5.2 Installation des Systems

1. Es sind alle benötigten, noch nicht auf dem System vorhandenen CPAN-Module zu installieren. Siehe dazu 2.3. Folgender Befehl nimmt hierbei alle Arbeit ab:

   ```
   $ perl -MCPAN -eshell
   Terminal does not support AddHistory.

   cpan shell -- CPAN exploration and modules installation (v1.7601)
   ReadLine support available (try 'install Bundle::CPAN')

   cpan> install Modul::Name # für jedes Modul
   ```

2. Der Webserver muss so eingestellt werden, dass er symbolische Links auf ausführbare Dateien erlaubt, zumindest unter der Bedingung, dass die Benutzer-ID des Symlinks mit der der referenzierten ausführbaren Datei übereinstimmt. Nähere Informationen dazu in der Dokumentation des Webservers.

3. Entpacken des Programmarchivs im gewünschten Verzeichnis (z. B. `cgi-bin/` des Webservers) mit dem Befehl

   ```
   $ tar -xzf GatherYSe-0.1.tar.gz
   ```

   Ein Unterverzeichnis namens `GatherYSe/` wird automatisch angelegt.

4. Es ist sicherzustellen, dass die Dateien `run.pl` und `test.pl` vom Webserver ausgeführt werden können. Hierzu wird folgendes Shellkommando empfohlen (`www` muss ggf. durch die tatsächliche Gruppe des Webservers ersetzt werden). Selbstverständlich muss der Webserver alle Dateien und Verzeichnisse lesen dürfen.

   ```
   GatherYSe$ for i in {run,test}.pl; \
   > do chgrp www $i && chown g+x $i; \
   > done
   ```

5. Ebenso muss die Logdatei (s. GysDiag.pm, Zeile 36), durch den Webserver angelegt bzw. geschrieben werden dürfen. Es empfiehlt sich dazu der Einfachheit halber, sie im Log-Verzeichnis des Webservers abzulegen.

6. Anlegen der Anbindungskonfigurationen unterhalb von `configs/`. Die Konfigurationsdateien für das Angebot der Humboldt-Universität zu Berlin an E-Book-Kollektionen mit Volltextsuche liegen auf der CD im Unterverzeichnis `configs-hub/`.

7. Ausführung des Skriptes `configure.pl`. Dieses muss ohne Fehler durchlaufen, damit der nächste und letzte Schritt angegangen werden kann.

8. Schließlich sind im Administrationsbereich des Metalib-Portals (s. Kap. 4) die Quellen zu konfigurieren und mit dem External Hook für SRU-Quellen zu assoziieren. Details hierzu sind auch in der Metalib-Dokumentation zu finden. Im Host-Feld des Subscription Tabs ist jeweils einzutragen: `http://.../GatherYSe/run/kurzname?`, wobei „...“ durch die Hostadresse des Servers plus dem Verzeichnispfad zu GatherYSe zu ersetzen.

Literaturverzeichnis

[Con06] Conway, Damian: *Perl Best Practices*. Dt. Ausgabe der 3. Aufl. O'Reilly, 2006. – ISBN 3-89721-454-7

[CPA] *Comprehensive Perl Archive Network*. – URL: http://www.cpan.org. Letzter Zugriff: 28.07.2007

[Edu] Eduserv Technologies Ltd: *How Athens Works*. – URL: `http://www.athensams.net/how_athens_works`. Letzter Zugriff: 28.07.2007

[FBMP07] Foust, Jill E. ; Bergen, Phillip ; Maxeiner, Gretchen L. ; Pawlowski, Peter N.: Improving e-book access via a library-developed full-text search tool. In: *Journal of the Medical Library Association* 95 (2007), Januar, Nr. 1

[Les05] Lester, Andy: *WWW::Mechanize::FAQ*. 2005. – URL: `http://search.cpan.org/dist/WWW-Mechanize/lib/WWW/Mechanize/FAQ.pod`. Letzter Zugriff: 28.07.2007

[Lib04] Library of Congress: *SRU: Search/Retrieve via URL*. 2004. – URL: `http://www.loc.gov/standards/sru/`. Letzter Zugriff: 30.07.2007

[Lud07] Ludwig, Katja: *Organisation der Integration einer verlagsübergreifenden Suche in E-Books in ein lokales Bibliotheksportal*. Fachhochschule Potsdam, Fachbereich Informationswissenschaften, Diplomarbeit, Juli 2007

[Nat] National Library of Finland: *External Search Programs for Metalib*. – URL: `http://www.lib.helsinki.fi/finelib/english/nelli/ml_externals/`. Letzter Zugriff: 29.07.2007

[Ope] OpenQA: *OpenQA: Selenium*. – URL: `http://www.openqa.org/selenium/`. Letzter Zugriff: 28.07.2007

[PJ03] Powell, Andy ; Johnston, Pete: *Guidelines for implementing Dublin-Core in XML*. 2003. – URL: `http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/`. Letzter Zugriff: 28.07.2007

[RVRK05] Renner, Thomas ; Vetter, Michael ; Rex, Sascha ; Kett, Holger: *Open Source Software: Einsatzpotenziale und Wirtschaftlichkeit*. Fraunhofer IRB Verlag, Stuttgart, 2005. – ISBN 3-8167-7008-8. URL: `http://www.e-business.iao.fraunhofer.de/docs/fhg_oss-studie.pdf`. Letzter Zugriff: 29.07.2007

[Sch07] Schneider, Wolfram: *E-Mail an Florian Heß vom 18. Juli.* 2007

[Vö03] Völkel, Max: *Extraktion von XML aus HTML-Seiten. Das WYSIWYG-Werkzeug d2c.* Universität Karlsruhe (TH), Fakultät für Informatik, Diplomarbeit, Mai 2003

[Vö07] Völkel, Max: *E-Mail an Florian Heß vom 8. Mai.* 2007

[WCO01] Wall, Larry ; Christiansen, Tom ; Orwant, Jon: *Programmieren mit Perl.* Deutsche Ausg. der 3. Aufl. O'Reilly, 2001. – ISBN 3-89721-144-0

[WW98] Wolf, Misha ; Wicksteed, Charles: *Date and Time Formats.* 1998. – URL: `http://www.w3.org/TR/NOTE-datetime`. Letzter Zugriff: 31.07.2007

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine als die von mir angegebenen Quellen und Hilfsmittel verwendet habe. Diese Arbeit hat keiner anderen Prüfungsbehörde vorgelegen.

.....
Florian Heß
Berlin, d. 31.07.2007