

# An Agent-Based Architecture for Supporting High-Level Search Activities in Federated Digital Libraries

Norbert Fuhr\*      Norbert Gövert      Claus-Peter Klas  
University of Dortmund, Germany

## Abstract

Federated digital libraries offer a variety of information structures. In order to allow for different kinds of searches as well as for navigation along explicit and implicit links, a large set of searching and browsing functions has to be provided. However, non-expert users will need additional help for performing effective searches. In order to guide the user, we support strategic search activities at different abstraction levels (strategies, stratagems, tactics), which are structured in a hierarchy. For implementing these concepts, we have developed an agent-based architecture for accessing a federated digital library. The resulting system integrates searching and browsing, supports high-level search activities, allows for horizontal and vertical integration of services and is flexible and extensible.

## 1 Introduction

Most of today's digital libraries (DLs) are stand-alone information systems containing documents (or document metadata) in electronic form. In many areas, DLs still cover only a small fraction of the relevant literature, so users have to combine information from both traditional libraries and DLs. However, as the number of DLs increases, their overall coverage of the literature in certain areas becomes very high, and so many information searches do no longer require the involvement of traditional libraries. In this situation, dealing with numerous stand-alone DLs becomes unacceptable for the user: In order to satisfy an information need, s/he has to visit several DLs; searches have to be reformulated for each DL; browsing is possible to a limited extent within single DLs only; data about the same document is spread over different DLs like e.g. bibliographic, reference, citation and full-text databases

In order to overcome this situation, the concept of federated DLs has been proposed. Such a system should allow for searching and browsing across different DLs and merge the results for identical documents, thus giving the user the view of a single virtual digital library. However, different DLs do not only vary in the sets of

documents and the kinds of metadata, they also use different schemas for representing this information. In order to deal with the rich, and heterogeneous information structure of federated DLs, database-oriented approaches provide appropriate data structures and data types, but fall short in handling the intrinsic vagueness and imprecision of information searches in DLs. In contrast, information retrieval approaches focus on these aspects in combination with distributed searching, but do not consider the underlying information structures. Hypertext approaches cover mainly the browsing part of information seeking.

Obviously, a combination of these three different approaches is required in order to deal with federated DLs: The information structure should be modeled based on database concepts, and should be taken into account by the information retrieval methods for searching the DL. In addition, appropriate browsing mechanisms based on hypertext concepts should be provided.

An integration of these three approaches yields a rich set of searching and browsing functions. However, this complex functionality poses problems for non-expert users, since all functions are only low-level, and there is no guidance for following any search strategy. In contrast, [Bates 90] describes different levels of search activities which are based on empirical studies of the information seeking behavior of experienced users. Whereas typical information systems only support low-level search functions (so-called moves), Bates introduces three additional levels of strategic search functions (tactics, stratagems and moves) and notes the absence of system support for these functions.

In this paper, we present an agent-based architecture for user-oriented access to federated digital libraries. The user view on the underlying information structure is modeled as multi-level hypertext, with integrated searching and browsing. In addition, high-level search activities are implemented as cooperating agents. This architecture allows for the integration of different DLs and services and it can be easily extended by new functions and data sources.

In the remainder of this paper, we first discuss related work on user-friendly access to federated DLs. Then we describe the information structure in terms of multi-level hypertext, and we show how searching and browsing in a federated DL can be realized. After a brief review of Bates' ideas, we present the agent-based system

---

\*University of Dortmund, Computer Science 6, 44221 Dortmund, Germany. Email: fuhr@cs.uni-dortmund.de, fon: +49 231 755 2045, fax: +49 231 755 2405

design, and we explain the implementation of high-level search activities.

## 2 User-friendly access to a federated digital library

In order to be practically useful, a federated digital library should contain a significant portion of the literature a typical user requires for satisfying his information needs. For our work, we created a test-bed for the area of computer science comprising about a dozen digital libraries of different types: Tables of contents (of journals and conference proceedings), reference databases (like e.g. INSPEC), citation databases (e.g. CiteSeer) and full-text databases.

From a user's point of view, such a federation of digital libraries should be presented as a single virtual digital library, providing not only a single user interface and uniform data structures, but also inhibiting the borders between the databases involved. Retrieval in distributed collections is an issue addressed by many researchers, dealing e.g. with database selection ([Gravano et al. 94], [Fuhr 99a]), collection fusion ([Gauch et al. 96], [Callan et al. 95], [Voorhees et al. 95], [Baumgarten 99]) and heterogeneity w.r.t. database schemas and search predicates ([Fuhr 99b], [Chang et al. 99]). Architectures for federated DLs are presented e.g. in [Röscheisen et al. 98] and [Frew et al. 98]. However, all these approaches are system-oriented and do not support strategic search activities.

In contrast, the DLITE system presented in [Cousins et al. 97] takes a user-oriented approach and implements a wide variety of DL activities based on the workspace metaphor; however, only low-level of search activities are implemented in this system, and browsing is not well supported. The SFX approach for context-dependent document linking presented in [Sompel & Hochstenbach 99] offers a wide variety of browsing functions at the document level comparable to the *evaluate document* stratagem described below.

In our approach, we also take the more user-oriented perspective. Our focus is on search activities, for which we want to offer a broad range of functions in a structured way.

In the following, we first describe the integrated hypertext and retrieval system, which forms the basis for the implementation of high-level search activities. After giving a brief outline of Bates' ideas concerning this point, we first describe the agent-based architecture, and then we show how Bates' concepts are implemented in our system.

## 3 Functionality of a federated digital library system

### 3.1 Searching and browsing

As described above, most research in digital libraries has focused on distributed search, whereas browsing in a federated environment has hardly been considered so far. In a simplistic view, one could assume that all links are explicit, so navigating along these links would not be a research issue. However, in most of today's DLs, only few links are explicit, whereas the majority is implicit. Therefore, additional effort is necessary in order to allow for navigation along these links.

Following the idea of multi-level hypertext ([Agosti et al. 91]), we can distinguish four levels in DLs:

1. The *schema* level describes the schema of the DL (e.g. Dublin Core, MARC, BibTeX), which is at least a list of attributes (author, editor, title, year, ...). For rich bibliographic schemas like e.g. MARC, appropriate browsing facilities are required.
2. The *attribute value* level contains the values of the different attributes, e.g. the list of author names or for taxonomies (thesauri, classification schemes) the entries and their relationships (broader / narrower / related term).
3. The *metadata* level comprises the metadata records of the documents, i.e. the values of the bibliographic attributes as well as the content-describing attributes such as the abstract or classification codes.
4. The *full-text* level contains the documents themselves, usually in formatted form such as PDF or Postscript.

In the following, we will ignore the schema level, by assuming that there is one fixed schema (we use a slightly extended version of the BIBTEX attribute set [Lamport 86, pp. 139–147] for this purpose). Furthermore, due to the formatted form of the full-text documents, we consider these as atomic entities, which can only be the target of links.

In this multi-level hypertext, we have a variety of link types, namely aggregation links (between aggregations and their elements, e.g. journal issue – articles), citation links, hierarchical links (e.g. in classification schemes), attribute links (between an attribute value (e.g. an author name) and documents with this value), similarity links (between similar attribute values or documents) and fulltext links (between the bibliographic entry and the fulltext of a document). All these links are bidirectional, i.e. it should be possible to follow them in both directions. Attribute and full-text links connect the different levels, whereas aggregation, citation and similarity links are intra-level links.

The current systems offers very poor support for these links—even links within a single database are

only partially provided; moreover, links between different databases are not supported at all.

Thus, our approach for allowing navigation along all these links does not only increase the functionality of a single DL. By linking across different databases, we increase the value of the different sources, so that we get a synergy. In addition to these low-level searching and browsing functions, we also want to provide high-level operations that support the user's strategic thinking during a search. For this purpose, we adopt Bates' ideas of high-level search activities.

### 3.2 High-level search activities

In [Bates 90], four different levels of search activities are distinguished:

1. A *move* is an identifiable thought or action that is a part of information searching. Moves correspond to commands that are provided by today's IR or hypertext systems (e.g. enter a term, follow a link).
2. A *tactic* is one or a handful of moves made to further a search; thus, it already contains a strategic element. For example, breaking down a complex information need into subproblems, broadening or narrowing a query are typical tactics applied frequently. In [Bates 90], different kinds of tactics referring to monitoring, file structure, search formulation, term selection and ideas are described in detail.
3. A *stratagem* is a complex set of actions (comprising different tactics) exercised on a single domain (e.g. citation database, journal TOC). Often, an information need can be satisfied by a single stratagem.
4. A *strategy* comprises a complete plan for satisfying an information need. Thus, it typically consists of more than one stratagem (e.g. perform a subject search, browse through relevant journals and then find the documents cited by the most important articles).

Strategies are very much application-dependent. For stratagems, Bates mentions the following possibilities:

**Subject search:** Given some terms describing the topic of the search, one looks for documents containing these and/or related terms.

**Journal run:** Having identified a relevant journal, one browses through the tables of contents.

**Citation search:** Starting with a relevant document, one looks for other articles citing or cited by this document.

**Area scan:** After locating a subject area of interest in a classification scheme or thesaurus, one browses documents in this area.

**Author search:** Knowing an author relevant to one's topic of interest, one searches for publications from this author; also frequent co-authors are identified and searched upon.

Tactics are described along with the corresponding stratagems in section 5.3.

As an additional paradigm for guiding the search, Bates has proposed the berry-picking [Bates 89] concept: After the user has started with a preliminary description of his information need (expressed as initial query), s/he may find terms that describe the information need in a better way during the search. So these terms are added to the query, whereas initial terms may become obsolete.

## 4 Agent-based architecture

In order to implement high-level search activities, we choose an agent-based architecture (ABA) (see e.g. [Wooldridge & Jennings 95]). The following features of agents are relevant for IR applications (see also [Wondergem et al. 97]):

**Autonomy:** An agent is a process of its own, and thus it can operate independent of other agents.

**Intelligence:** An agent is able to process knowledge and to draw inferences; in our case of an IR application, an agent should be capable of uncertain reasoning.

**Reactiveness:** An agent reacts when prompted by another agent.

**Proactiveness:** An agent is able to take the initiative itself, e.g. when it detects changes in its environment that require action.

**Adaptiveness:** An agent can adapt its behavior to the application it is being used for.

**Communication:** An agent is able to communicate with other agents peer-to-peer.

ABAs for IR applications have been proposed and used by several other researchers already, e.g. for personal assistants [Lieberman 95], mediators [Arens et al. 96], Searchbots (Savvysearch<sup>1</sup>, Metacrawler<sup>2</sup>) or filtering [Moukas & Maes 98]. However, our feeling is that most of these approaches make poor use of the advantages of ABAs, mainly due to the fact that they implement only a rather limited set of retrieval and browsing functions (i.e. they support only one stratagem). In contrast, with the idea of high-level search activities, our approach is based on a rich set of functions at different abstraction levels, and thus it may profit to a greater extent from the advantages from ABAs. Another ABA for DLs has been sketched in [Atkins et al. 96], but this work does not address the issue of high-level search activities.

For our DL application, communication and the control flow (including autonomy, reactivity and proactiveness) are the most relevant features.

<sup>1</sup><http://www.savvysearch.com>

<sup>2</sup><http://www.metacrawler.com>

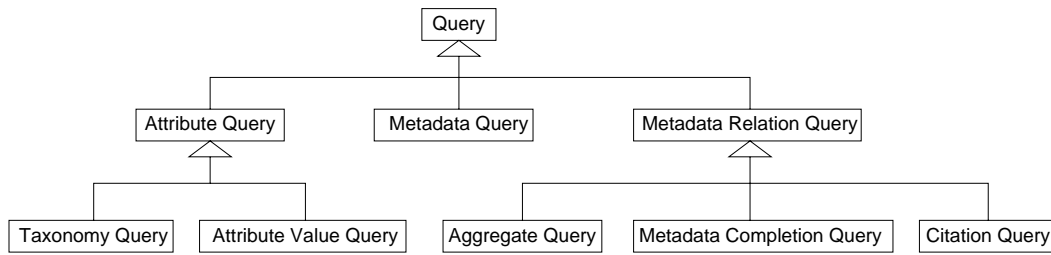


Figure 1: Query hierarchy

**Communication** As communication language, we use a small subset of KQML [Finin et al. 94] so far. Our knowledge representation consists of a few object classes and an ontology for functions. The content of objects is exchanged in XML format, based on DTDs for the corresponding classes.

In principle, there are two major types of content, namely queries and results. The different types of queries are depicted in the class diagram shown in figure 1. A query is either an attribute query, a metadata query or a metadata relation query. An attribute query asks for attribute values fulfilling a specified condition (e.g. author names where the last name is “Smith”); as a special case, a taxonomy query allows for searching and browsing in a thesaurus or classification scheme. Metadata queries search for (metadata records of) documents fulfilling the specified condition(s), where each condition is a triplet (attribute, predicate, value) (e.g. author = “Smith”). Queries are just sets of conditions, where a condition is either mandatory, optional or prohibitive. Metadata relation queries specify the metadata record of a document and one of the relations citing, cited, aggregate and complete. The first two search for documents citing or cited by the specified one, respectively. The aggregate relation retrieves publications from the same aggregation, i.e. journal issue or conference volume. Finally, complete serves for completing the given metadata record by searching other databases for metadata about the document specified (this may also include pointers to the full-text of the document).

Results are either lists of attribute values or lists of document records. The former are returned in response to attribute queries. In the latter, a document record is either a surrogate (in the form as it is returned from the corresponding DL in the result survey) or a metadata record with bibliographic and content-describing attributes.

Functions that can be performed by agents are organized in an ontology. By using a hierarchy of functions, an agent may either ask for a very specific function (which is provided by only one agent), or for more general functions, for which several agents with different implementations are available. For example, in order to refine a query, an agent may invoke the function `service/refine-query`, or

it may ask for a specific method of query refinement, e.g. `service/refine-query/thesaurus` or `service/refine-query/associations`, invoking either a thesaurus-based method or a service using statistical term associations (as they are provided by some Web search engines for refining a query). In a similar way, `wrapper` addresses the set of all wrapper agents, whereas `wrapper/cora` invokes the wrapper agent of the CORA DL only.

**Control flow** For realizing a specific high-level search activity, there may be more than one possible approach. For example, in order to refine a query, one agent might perform a thesaurus lookup in order to locate narrower terms; alternatively, another agent could consider term associations (as they are provided by some Web search engines for refining a query). In a similar way, more than one DL may contain relevant answers for a given query. Thus, there are two issues that have to be solved: First, all possibilities for solving a specific (sub)problem have to be taken into account, and then the most promising among these should be performed. As a major goal, we are aiming at an extensible system where new methods for solving specific problems can be added to the system without great effort. Thus, any centralized control would be a hindrance on achieving extensibility. For this reason, we adopt the concept of contract networks ([Davis & Smith 83]) in order to cope with the two issues mentioned before. In a contract network, the agent formulating a task (a manager) broadcasts this task to other agents; any agent (contractor) who is willing to fulfill this task replies with a bid for this task. Then the manager selects one or more contractors with the best bids and establishes a contract with these agents. Here manager and contractor are roles of agents with respect to a specific task; for different tasks, the same agents may have just the opposite role. In the current implementation, we use a simplified version of contract nets only, which mainly deals with the problem of unpredictable response times of DLs: Following a request from a manager, any contractor willing to fulfill this task sends an acknowledgement message and then immediately starts working. The manager first collects the acknowledgements that arrive within a predefined short time period, and

then waits for the results of the corresponding contractors (at most up to a timeout limit). Thus, usually short response times can be achieved, and wrappers that have lost contact to their DL are not considered.

**Visualization** The user interface is managed by a user interface agent, based on the model-view-controller (MVC) concept. Corresponding to the partition of the user interface into several tiling windows (or frames), there are MVC tuples for each window. Here the model keeps the corresponding state of the application, the view presents this state in the corresponding region on the screen, and the controller accepts the user's input and notifies the model about the necessary actions. After updating its internal state, the model prompts the view for visualizing the new state.

In our system, the system state corresponds to the combination of the standard objects described in the communications paragraph above: The current (metadata) query, attribute values (returned in response to an attribute query), the current result list (of surrogates), and the metadata record(s) currently focused. Thus, by assigning an MVC tuple to each of these objects, the system state becomes transparent to the user. (Also, on the system side, other agents may query the models about their current state.) This architecture is still very flexible with respect to the actual visualization, since views and controllers can be implemented in different ways.

User guidance is based on strategies and stratagems. A strategy determines the possible sequence of stratagems. There is always one stratagem agent which is active, and any stratagem restricts the set of actions (i.e. tactics and moves) that can be performed at a certain point of the dialog. In order to visualize this concept, the "Select stratagem" window on the left of figure 2 displays the menu of possible stratagems, where the active stratagem is highlighted. At any point in time, users may switch between stratagems, which affects mainly the content of the query and the attribute value window. Tactics are presented as buttons or links in red, starting with a capital letter. In contrast, moves correspond to a variety of interactions such as typing in a field, marking items displayed or invoking low-level functions such as adding marked values to the current query or submitting a query.

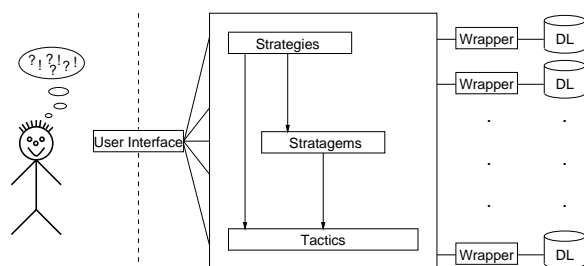


Figure 3: Overall architecture

## 5 Implementing high-level search activities

Figure 3 gives a rough sketch of the overall architecture. The user contacts the system via the user interface agent, which in turn has access to the agents implementing strategies, stratagems and tactics. On the other side, there is a wrapper agent for each DL, thus providing a standard interface for the remainder of the agent system. In the following, we describe the components of the system in detail.

### 5.1 User interface

In order to develop a user-oriented system with a new set of high-level operations, we first had to make certain decisions about the interaction between the system and the user. Since our current focus is mainly on functionality (whereas visualization will be considered in more detail later), we assumed a simple structure of the user interface, in combination with certain restrictions on the dialog between human and machine.

The interface consists of the following windows:

**Query:** The query window presents the current query.

A query consists of a set of conditions, where each condition has a connector specifying whether the condition is mandatory, optional or prohibitive. The user can add new conditions to the query, but the attribute the condition refers to may be restricted by the stratagem currently in action (see below). In order to implement berry-picking, each mandatory condition has an age (measuring the number of dialog steps since its addition to the query); the berry-picking tactics turns the oldest conditions into obsolete ones. All obsolete conditions are shown in a sub-window, from which the user may bring them back to the current query (with zero age).

**Attribute values:** This window allows for searching and browsing attribute values. Depending on the structure of the attribute, the values are presented either as a sequential list (e.g. for author names or journal titles) or as a tree (e.g. for classifications or aggregations). Users may specify a search condition for values of a specific attribute (again, the attribute may be restricted by the stratagem) or browse through the values displayed. Selected values can be added as conditions to the query.

**Result list:** After submitting a (metadata) query, the result is displayed as a list of surrogates in the result window. Entries from this list can be selected for being displayed as metadata records in the document window. The result set can be modified by specifying filter conditions (formal criteria like e.g. publication date or document type). In addition, it is possible to mark documents as (non)relevant for performing relevance feedback, thus generating a new query.

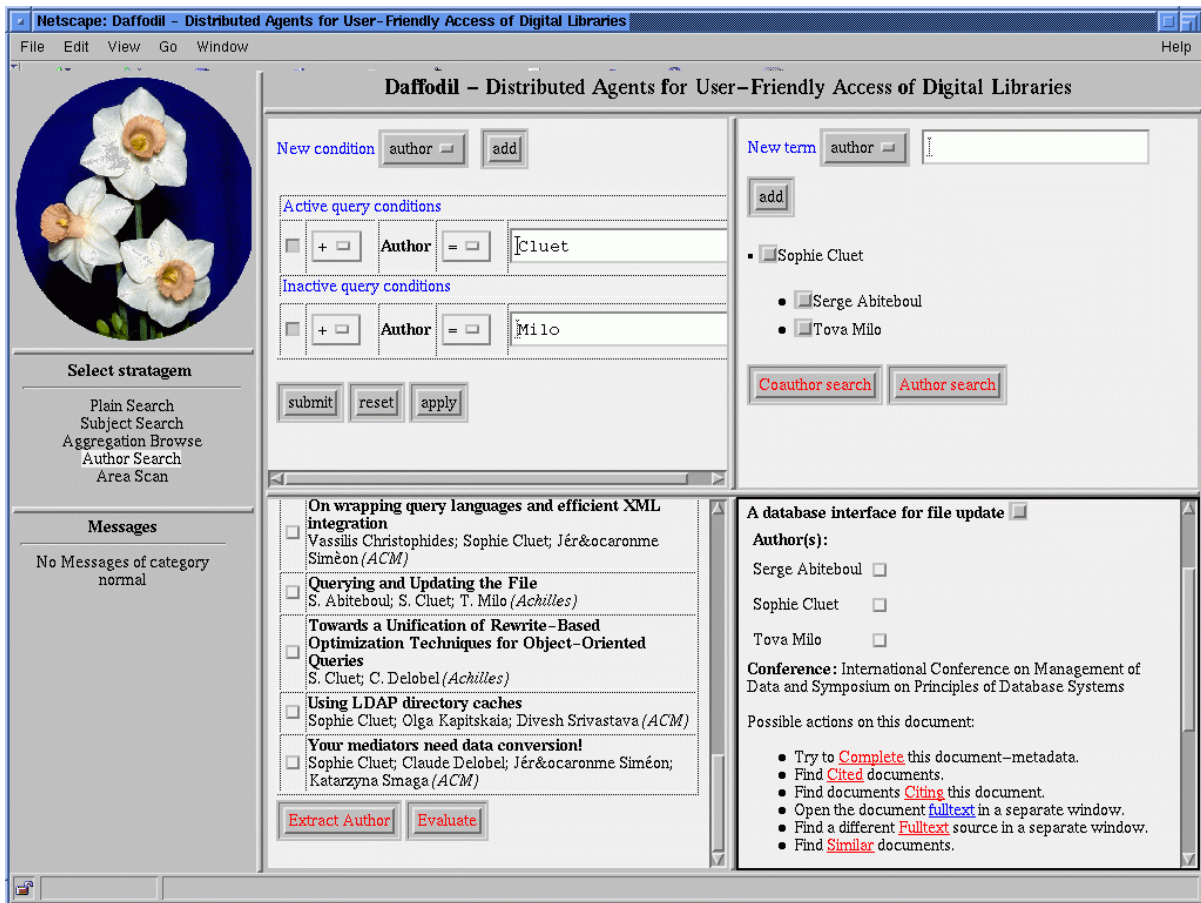


Figure 2: User interface: results of a (co)author search

**Document:** The document window shows one or more metadata records of documents along with a rich set of possible actions. Here the following tactics can be invoked by the user:

- *Complete metadata* asks for more complete metadata information about a document. For this purpose, other DLs (than the one producing this entry) are searched for information about the current document, and then the data is merged.
- *View full-text* opens a new browser window where either the full-text of the document, the results of an appropriate search in public servers (e.g. in CORA<sup>3</sup> or CiteSeer<sup>4</sup>) or an appropriate ordering form are displayed.
- *Aggregation search* looks for articles from the same volume (in case the document is a journal or proceedings article).
- *Cited by* searches for publications citing this document.
- *Cites* retrieves the publications cited in the current document.

- *Similarity search* looks for similar documents (based on title and abstract).
- *Mark attribute value*, followed by *pick value*, picks an attribute value from the current document as berry for the berry-picking tactics mentioned above.
- *Mark attribute value*, along with *ad-hoc query* performs a search with the given attribute value alone.

**Messages:** Here the system gives information about the progress in processing the current user request. Most of the time, wrappers are waiting for the responses from the DLs, so messages about the status of these interactions give appropriate feedback to the user while s/he is waiting for the overall result.

In the following, we first describe the standard search strategy implemented by our system, and then we present the available stratagem along with the tactics employed.

## 5.2 Strategy

Since strategies are highly application dependent, we spent little effort on this type of search activities. As

<sup>3</sup><http://www.cora.whizbang.com/>

<sup>4</sup><http://www.csindex.com/>

a standard strategy, we provide a pre-search interview that asks the user for known facts about the items to be searched. Depending on the answers to these queries, the system proposes one or more initial stratagems:

- Do you know any relevant documents?  
Based on a known document, a citation or similarity search can be performed; for locating this document, the *plain search* stratagem is proposed.
- Do you know authors publishing in this area?  
With an author name known, the *author search* stratagem can be employed.
- Do you know any relevant journals or conferences dealing with this topic?  
Here an *aggregation browse* on journal or conference volumes is applicable.
- Can you give a few terms describing your topic?  
With the terms given, a *subject search* can be performed.

In case none of these questions is answered positively, the system proposes an *area scan* as initial stratagem.

### 5.3 Stratagems

A stratagem is a complex set of actions exploiting a single domain. In the following, we describe the stratagems of our system, along with the tactics employed.

**Author search** searches for publications by focusing on authorship. Besides entering the author names directly, a user may search or browse the author index. From a retrieved set of documents, the author names can be extracted on demand. As additional tactics, our system can search for coworkers of a given author. Currently, this goal is achieved by searching for coauthor. An alternative method would be the use of affiliation information—however, this data is not available in the present system. Except when author names are entered directly, all tactics provide lists of authors from which the user may select appropriate entries to become part of the query; for each selected value, also the search priority (mandatory, optional or prohibitive) has to be specified.

**Subject search** is based on a topic description (by means of a set of terms) given by the user. As basic functions, this stratagem provides operations similar to *author search*: entering terms directly, searching or browsing term lists or extracting terms from a result set. The coworker tactics corresponds to the search for related terms of a given set of terms; as a first implementation of this strategy, we use the “refine query” function of a Web search engine. (Due to the agent-based architecture, adding alternative methods is straightforward). For a single term selected from the resulting term lists (or entered directly), the system can propose related, narrower or broader terms. There are three different implementations of this tactics, which use the INSPEC thesaurus, the ACM Computing Classification System

or the refine function of a Web search engine, respectively.

**Plain search** mainly serves for locating a document known by the user. Thus, the user may select arbitrary attributes for formulating the corresponding query. For each search condition, the user first selects an attribute, and then either enters the search value directly or browses/searches through the corresponding database indexes and then selects the preferred values.

**Area scan** supports navigation through a classification scheme (in our case the ACM Computing Classification System), which also provides links to the documents belonging to each category. The user may either search for a specific entry or start browsing from the top level. After selecting one of the entries resulting from this step, s/he may navigate to related, broader or narrower concepts. Having found an appropriate entry, one can search for appropriate documents by changing to the *evaluate result* stratagem. Since not all DLs support this classification scheme, a classification entry also is translated into a free text search; currently, we only use the corresponding terms from the classification scheme; a better solution planned for the future is the use of statistical associations between classes and text terms which can be derived from a dataset for which both types of information are available.

**Aggregation browse** focuses on tables of contents of journals or conferences/conference series. Titles of aggregations can be searched and browsed in the usual way. Alternatively, the system can extract titles (and volume numbers) from a given search result. From the titles displayed, the user may choose one in order to view the list of volumes available. By picking one of the volumes, its contents are displayed by means of the *evaluate result* stratagem. As a shortcut, the user may pick a (title, volume)-pair from the extracted list.

### 5.4 Status of implementation

The system as described before is completely implemented. Currently, there are wrapper agents for 12 different DLs, 24 tactics agents, 7 stratagem agents and a user interface agent. The user interface is implemented by using HTML with forms and frames, thus it can be easily accessed via a Web browser. On the server side, a CGI script is started, which connects to the user interface agent, from where the other agents are involved as needed. For later versions of the user interface, we plan to use Java applets, thus allowing for more flexibility in the user interface.

## 6 Conclusions and outlook

In this paper, we have shown that federated DLs require a rich set of searching and browsing functions at different abstraction levels. For this purpose, our system offers the following features:

- By combining metadata from different DLs, the user gets all the information available for each document displayed.
- Using explicit and implicit linking information from the various sources, we implement a rich set of browsing possibilities in the form of multi-level hypertext.
- For iterative query formulation, the berry-picking paradigm is supported by our system.
- In addition to the low-level functions offered by the underlying DL systems, we also implement strategic search activities.
- The higher-level search activities are organized in a hierarchical structure, thus providing better user guidance.

Most of these features are lacking in comparable systems, and there is no system integrating all these functions.

The current version of our system implements the integration of searching and browsing, the support of high-level search activities, horizontal and vertical integration of services and is flexible and extensible. For visualization, we have chosen a simple solution for the first prototype.

## References

- Agosti, M.; Colotti, R.; Gradenigo, G.** (1991). A two-level hypertext retrieval model for legal data. In: *Proc. SIGIR*, pages 316–325. ACM.
- Arens, Y.; Knoblock, C.; Hsu, C.** (1996). *Query Processing in the SIMS Information Mediator*. Austin Tate, AAAI Press, Menlo Park, CA. <http://www.isi.edu/sims/papers/96-arpi-book.ps>.
- Atkins, D. E. et al.** (1996). Toward Inquiry-Based Education Through Interacting Software Agents. *IEEE Computer* 29(5), pages 69–76.
- Bates, M. J.** (1989). The design of browsing and berrypicking techniques for the online search interface. *Online review* 13(5), pages 407–424.
- Bates, M. J.** (1990). Where Should the Person Stop and the Information Search Interface Start? *Inf. Proc. & Mgmt.* 26(5), pages 575–591.
- Baumgarten, C.** (1999). A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval. In: *Proc. SIGIR*, pages 246–253. ACM.
- Callan, J.; Lu, Z.; Croft, W.** (1995). Searching Distributed Collections with Inference Networks. In *Proc. SIGIR*, pages 21–29. ACM.
- Chang, C.; Garcia-Molina, H.; Paepcke, A.** (1999). Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. *ACM TOIS* 17(1), pages 1–39.
- Cousins, S.; Paepcke, A.; Winograd, T.; Bier, E.; Pier, K.** (1997). The Digital Library Integrated Task Environment (DLITE). In: *Proc. ACM Digital Libraries*, pages 142–151. ACM.
- Davis, R.; Smith, R. G.** (1983). Negotiation as a Metaphor for Distributed Problem solving. *Artificial Intelligence* 20(1), pages 63–109.
- Finin, T.; Fritzson, R.; McKay, D.; McEntire, R.** (1994). KQML as an agent communication language. In: *Proc. CIKM*, pages 456–463. ACM.
- Frew, J. et al.** (1998). The Alexandria Digital Library Architecture. In: *Proc. ECDL*, pages 61–74. Springer.
- Fuhr, N.** (1999a). A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM TOIS* 17(3), pages 229–249.
- Fuhr, N.** (1999b). Towards Data Abstraction in Networked Information Retrieval Systems. *Inf. Proc. & Mgmt.* 35(2), pages 101–119.
- Gauch, S.; Wang, G.; Gomez, M.** (1996). ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines. *J. of Universal Computer Science* 2(9).
- Gravano, L.; Garcia-Molina, H.; Tomasic, A.** (1994). The Effectiveness of GLOSS for the Text Database Discovery Problem. In: *Proc. SIGMOD*, pages 126–137. ACM.
- Lampert, L.** (1986). *LaTeX - A Document Preparation System*. Addison-Wesley, Reading, Mass.
- Lieberman, H.** (1995). Letizia: An Agent That Assists Web Browsing. In: *Proc. IJCAI*.
- Moukas, A.; Maes, P.** (1998). Amalthea: An Evolving Multiagent Information Filtering and Discovery System for the WWW. *J. Autonomous Agents and Multi-Agent Systems* 1.
- Röscheisen, M.; Baldonado, M.; Chang, K.; Gravano, L.; Ketchpel, S.; Paepcke, A.** (1998). The Stanford InfoBus and Its Service Layers: Augmenting the Internet with Higher-Level Information Management Protocols. In: *Digital Libraries in Computer Science: The MeDoc Approach*. Springer.
- Van de Sompel, H.; Hochstenbach, P.** (1999). Reference Linking in a Hybrid Library Environment. Part 2: SFX, a Generic Linking Solution. *D-Lib Magazine* 5(4). [http://www.dlib.org/dlib/april99/van\\_de\\_sompel/04van\\_de\\_sompel-pt2.html](http://www.dlib.org/dlib/april99/van_de_sompel/04van_de_sompel-pt2.html).
- Voorhees, E.; Gupta, N.; Johnson-Laird, B.** (1995). Learning Collection Fusion Strategies. In *Proc. SIGIR*, pages 172–179. ACM.
- Wondergem, B.; van Bommel, P.; Huibers, T.; van der Weide, T.** (1997). Towards an Agent Based Retrieval Engine. In: *Proc. BCS-IRSG Colloquium on IR Research*, pages 126–144. Robert Gordon University, Aberdeen.
- Wooldridge, M.; Jennings, N. (eds.)** (1995). *Intelligent Agents: Theories, Architectures, and Languages*. Springer.