

**System Components for Embedded Information Retrieval from  
Multiple Disparate Information Sources**

R. Rao, D. M. Russell and J. D. Mackinlay

UIR-R-1993-05



# System Components for Embedded Information Retrieval from Multiple Disparate Information Sources

Ramana Rao, Daniel M. Russell, and Jock D. Mackinlay  
Xerox Palo Alto Research Center  
3333 Coyote Hill Rd; Palo Alto, CA 94304  
<rao@parc.xerox.com>

## Abstract

Current information retrieval interfaces only address a small part of the reality of rich interactions amongst user, task, and information sources. We view information gathering as an interactive, iterative activity involving multiple disparate information sources and embedded in the context of broader processes of information use. We have developed two key system components that enable information workspaces that adhere to this reformulation of information retrieval. The first is a design for a user/system interaction model for retrieval from multiple, disparate information sources. The second is a repository modeling system, called Repo, that represents meta-information about different information repositories in a manner that supports system operation as well as provides a direct information resource to the user. To test these ideas, we have utilized Repo and embodied the interaction model in the user interface of a system called Labrador.

## Introduction

The last ten years have seen a proliferation of information sources available to users from desktop computers. In addition to the usual collections of documents created and managed by individuals and their coworkers, there are documents available from servers on enterprise and world-wide networks, most notably the Internet. Furthermore, there are an increasing number of commercial information distributors that provide access to bibliographic citations, newspaper and magazine articles, financial and business data, and much more. Unfortunately, user-centered technology to help cope with this increasingly complex information landscape has not kept pace. Most current work on information retrieval doesn't acknowledge the larger process of which information gathering is a part, nor does it deal with the harder

issues arising during the use of multiple, disparate information sources.

In previous work, we have developed information access systems that address both the larger context of information use and support for integration of distal information sources [4, 19]. In this paper, we present further work on refining and fusing the notions of information workspaces and retrieval infrastructure. In particular, we have developed two key system components for building information workspace with capabilities for retrieval from a multitude of disparate sources: an interaction model that is firmly grounded in our view of embedded information retrieval; and a repository model that plays two roles in multiple repository use, a system role in implementing access to different information sources, and an informational role in supporting the user in the use of the sources.

Our view of embedded information retrieval, which is presented in Section 2, was developed by observation and analysis of real information gathering by ourselves and others. This view has informed the design of an interaction model, by which we mean a formal description of a paradigm for interaction between a user and a system. Our interaction model characterizes the interaction between the user and the information workspace with built-in access to a multitude of information sources as depicted in Figure 1. The user manipulates objects in the information workspace to retrieve units of information from multiple repositories. The interaction model, presented in Section 3, enumerates the particular kinds of objects that populate the workspace and the actions the user can perform on them. Though, of course, this model was influenced by specific ideas regarding user interface, it is defined independently of user interface design and thus is of prescriptive value for building retrieval systems.

The second key component presented in this paper is a gateway subsystem that provides uniform access to multiple sources containing heterogeneous information and served by different local and distal retrieval systems of varied functionality. Such a system component must necessarily embody an understanding of the properties of the supported repositories and retrieval systems. Our approach has been to capture this understanding in forms that serve as a basis for system operation and also as an information resource in

---

<sup>0</sup>Reprinted From: Proceedings of 1993 ACM Symposium on User Interface Software and Technology, Atlanta, GA, November 1993, ACM SIGGRAPH and SIGCHI

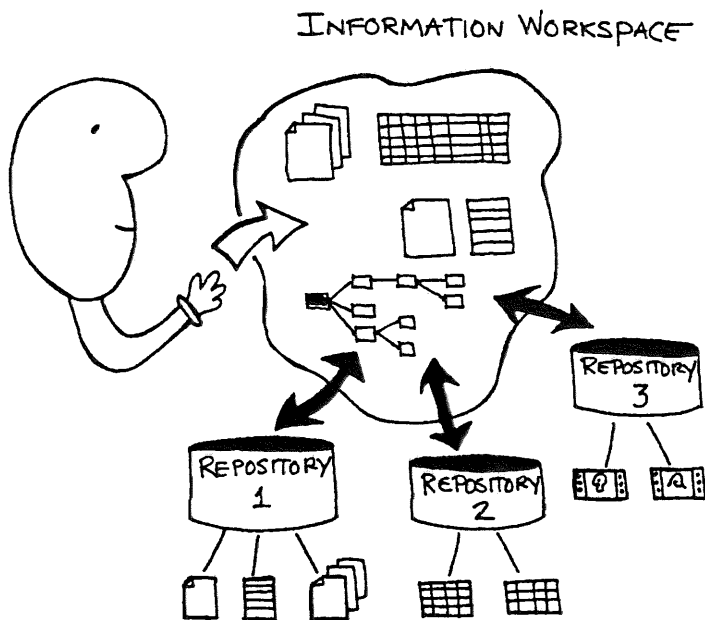


Figure 1: User in a Retrieval-Enabled Information Workspace

its own right. In Sections 4 and 5, we describe our positions on repository modeling and our implementation of system called Repo that embodies this position.

We have incarnated the interaction model in the user interface of a system called Labrador, described in Section 6, that provides a simple information workspace for retrieving, visualizing, and managing information from a variety of information sources. Labrador provides a concrete instance of embodying the interaction model and using Repo, though both are of greater applicability.

## 1 Related Work

The notion of interactive retrieval has a long tradition in both the information retrieval and user interface communities. Relevance feedback has been explored as a tool for improving retrieval performance since the late sixties [11]. User interfaces based on the notions of retrieval by criticizing examples or by reformulation were explored in [26, 10, 13]. These techniques are broadly believed to improve the ability of users to find information, and to some extent, validated through experimental, theoretical, and simulation studies [8, 22, 24, 25]. One of the observations in [8] is that relevance feedback is useful because it allows users to make a richer query than they would otherwise be able to make. Our interaction model, partially out of necessity in

dealing with disparate search engines, but also because of inherent utility, allows the query refinement and feedback process to include a broader range of user activities.

A number of systems have appeared in the last few years that allow users to locate information resources distributed on the Internet[23]. Three such systems, World Wide Web (WWW) [3], Gopher[15], and Prospero[16] are based on a navigational metaphor in which information resources are linked together into a hypertext or a hierarchical file structure. The navigational interaction model is based on a cycle of moving somewhere and browsing to see what's there and to decide where next to go. WWW, based on hypertext, assumes that information is intermixed with navigation; Gopher, based on a hierarchical file system metaphor, provides information nodes at the leaves of file system. The navigational interaction model is easy to use and understand, but breaks down in lots of situations involving large, complex, poorly-structured, multiuser, or variegated information sets.

The Wide Area Information Server (WAIS) [12] is another system for information access on the Internet. It is based on a retrieval-oriented interaction model using a textual similarity search and relevance feedback. The WAIS approach is based on standardizing the backend retrieval interface: textual information is indexed and served by a WAIS server with a single search operation using an extended version of a standard library retrieval protocol. Our work comes from the other end, the user's personal workspace. Rather than standardizing the backend, we attempt to model the diversity of the available sources and retrieval engines in the frontend and allow the user to exploit or finesse the heterogeneity of the information sources as necessary.

Gopher, WWW, and Propero all provide "gateways" to other Internet resource-finding systems of both navigational and retrieval-oriented interaction models (including WAIS). In each of these cases, the gateway is seen as a translator that integrates foreign information sources into the local world view. We provide a gateway to retrieval systems as well, though a much richer gateway since we try to incorporate distal functionality into our model in an abstracted form. In addition, we attempt to provide the understanding built into the gateway as a direct information resource to the user.

Erickson and Salomon, in [9], describe the investigation phase of work at Apple on building an interface to Dow Jone's DowQuest service which provides access to 350 news sources. Their formulation of the problem is closely aligned with our own. In particular, they, like we, advocate the importance of meta-information for supporting the selection of databases. In addition, based on their observations of general information users, they advocate support for broader information use activities like skimming, annotating, and organizing information.

The TeamInfo system is an experiment at HP Labs into building a "group memory" for sharing collected information resources among members of a workgroup [2]. Many issues regarding categorization for different purposes by dif-

ferent members of the group were exposed by this effort. Consequently, they also turned to search and visualization techniques for accessing information.

Paepcke, also at HP Labs, has explored how the collection of databases provided by Dialog could be made to look like a collection of objects in a single object-oriented database [17]. He is primarily interested in modeling the field structure of records in different databases and the relationships between the various fields and indices across databases. Many of his ideas are directly relevant to our work on repository modeling, though our own concerns are broader, since a broader range of meta-information is necessary for supporting the full span of user activities we wish to support.

## 2 Embedded Retrieval

One of our primary objectives in building information access systems is to support the style of interaction that is typical of real information gathering as it is embedded in actual work. Our understanding of this activity is supported by observations of our own research behavior and of a number of subjects engaged in information structuring work[21] as well as by analytical and observational studies by others[1, 5, 9, 14].

Conventional interfaces for information retrieval view retrieval as a self-contained task in which the user formulates a query against a single database to obtain matching documents. This view is technology-centered and misses the reality of user and task context in a number of significant ways. Information gathering is almost always in service of a larger work activity in which the patterns of gathering, examination, analysis, and, in general, use are richly intertwined. One specific consequence of this is that strategy decisions are often dominated by the overall purpose or task. In addition, users often deal with multiple information sources and associated retrieval services with differing characteristics of content, form, and functionality. In many cases, user activity is as much about understanding these characteristics as it is about finding particular documents.

The following scenario highlights the essential nature of real information gathering activity:

Joe fidgets, attempting to think about where in the world he should seek information for a presentation on object-oriented databases to his boss. After thinking about articles he'd read and some preliminary browsing of a conference proceeding, he eventually decides to call the Technical Information Center as he pulls down several years of the proceeding from his shelf. The librarian he talks to has some ideas, but doesn't seem too sure, and so Joe thanks her and hangs up the phone. As he looks through the proceedings, he finds some relevant articles that he tags with Postit notes, and

then he is reminded that he has some articles in his file cabinet. He opens and scans several drawers and then remembers that he had filed a survey article in his surveys file. He retrieves the article. Paging through it, he decides that this provides adequate information for him to start preparing his presentation. As he works, he returns to the proceedings to count articles on different topics, and to find papers on specific systems. He copies some pages from some of the articles and puts them in piles on his desk.

This scenario captures the iterative, interactive, back-and-forth nature of retrieval activity. Retrieval is embedded in and entwined with other kinds of information work. It is based on a wide variety of cues and insights that arise out of the interaction. It exploits the rich and variegated structure of information in the world.

## 3 Interaction Model

Based on the view of information retrieval presented in the last section, we have designed an interaction model. This model describes the interaction between the user and various objects that populate his information workspace. Our goal in this interaction model is to support the interactive nature of information retrieval where users reuse and reformulate queries and results. Search event objects, as shown in Figure 2, codify important retrieval events and provide a basis for supporting other system functionality including a history mechanism and visualizations.

We have fashioned a graphical notation, similar to that used in various object-oriented analysis methods (e.g. [20]), that depicts the essential classes of objects, their interrelationships, and the actions that the user can perform on them. Figure 2 uses this notation to capture our interaction model. The diagram can also be viewed as a *user conceptual model*, a specification of the concepts the user must understand to be able to effectively interact with the system.

The notation of Figure 2 doesn't capture how objects come into existence as a consequence of user action, or more generally, the dynamical nature of the interaction. The following account describes the interaction in greater detail. The ordering of items in this list is not necessarily required by the model. The account can be read as a walkthrough of the elements depicted in the figure with objects in bold and actions in italics on first mention:

- The user *creates* a **search event**, that he fills in by *setting its scope* and *specifying a query expression*. The *scope* determines the part of the available information universe to search which may be items from previous searches as well as **repositories** (a private, local, or public database) in entirety or in part. The details of specifying the querying can vary greatly depending on the scope.

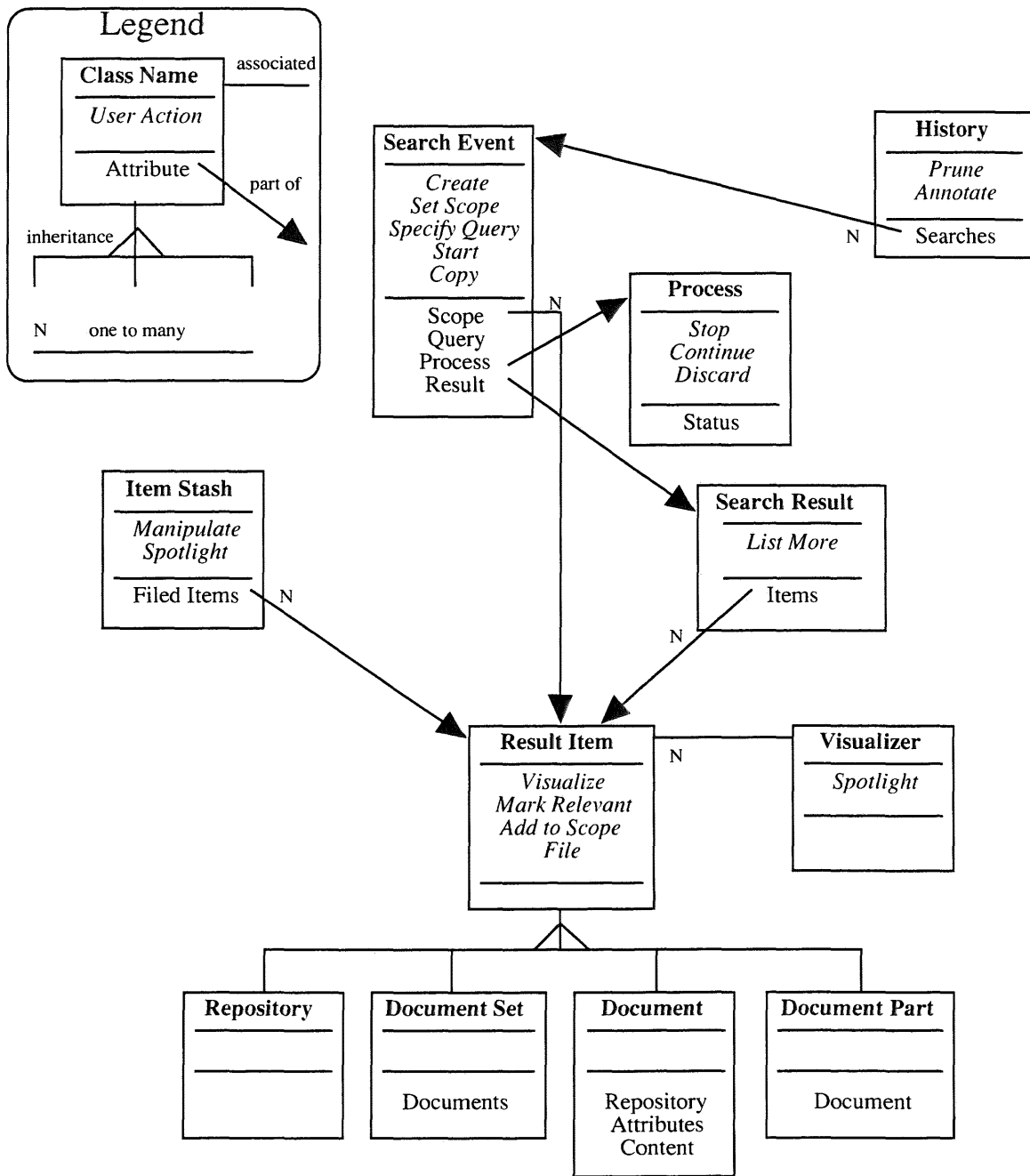


Figure 2: The rectangles represent classes of objects with which a user interacts in our framework. History and Item Stash are one instance classes, but all others classes will typically have many instances active at any given time. The lines between classes indicated the three kinds of relationships between classes: inheritance, part-of, and association. Note that all classes that inherit from “Result Item” support all of its user actions.

- An important kind of search is search for appropriate repositories, in essence, a meta-search. In this case, the scope may be all repositories, and the query specification is aimed at finding matching repositories, rather than documents.
- The user *starts* the search, which causes a **process** to be created. The user can *stop* the process, *continue* it, or *discard* it altogether. Once the search is started, it can no longer be changed, though the event can be *copied*. The copy can then be started as is or after modification.<sup>1</sup>
- The process creates a **search result** which contains a set of items that match the query. Since the process may take a long time or may be stopped manually by the user or automatically by the system (for example when there are many, many items that match), the result at any given point may only be partially completed. The user can force the result to *list more* of the matching items if the process has stopped.
- A persistent **history** of searches is maintained so that users can return to all search results or reuse previous queries. The history can be *annotated* to aid future use. This history can be automatically and/or manually *pruned* according to user needs and preferences. The history captures both chronological information and logical interrelationships between searches (e.g. and-refinement by using results as the scope of a further search or or-refinement by combining results into a scope).
- Depending on the kind of query that was specified, the **result items** in the result may be **document sets**, **documents**, **document parts** (i.e. pieces of documents), or **repositories**. The entire result or subsets of items in the result can be used in several ways. They can be used in further search: they can be used as *relevance* feedback to the system, which means that they are used as part of a new query specification, or alternatively, they can be used in the *scope* of a new search object. They can be individually *visualized* using one or more **visualizers**. Finally, they can be *filed* into a persistent **item stash**.
- Repositories or sets of result items may be visualized using one or more special **visualizers** suited to those kinds of objects. The various visualizers instances may be from radically different classes at one extreme or they may be slight view variants (e.g. scrolled, filtered) of the same class at the other extreme. Search may be used within a visualizer to *spotlight* specific items within the context of the larger visual structure.
- The user can *manipulate* the objects on the **item stash** to arrange them into new structures or new documents as necessary to support their work. This portion of the information workspace uses a direct manipulation, spatial/navigational metaphor to allow the user to organize and manage objects. In addition, search can be supported in the item stash as an auxiliary means for finding objects.

## 4 Repository Modeling

Providing effective access to repositories of widely varying types depends critically on modeling the range of variations across the types. This information is functionally necessary for integrating disparate repositories and retrieval system into a unified framework, but as importantly (though usually not supported) it can be directly useful to the end-user in selecting and understanding repositories and the nature of their contents.

Our position is motivated by the practices of professional searchers who use a multitude of online sources. For example, Erickson and Salomon [9] observed that a particular set of expert online searchers spent a remarkable amount of time at their weekly status meeting sharing information about databases: topics included newly available database, information quality, frequency of updates, timeliness of updates, costs, as well as situations in which a particular database should be consulted. The importance of meta-information also constantly came up in our discussions with a professional searcher on the PARC Information Center staff as well as with customer service representatives and information specialists from Dialog, a major online database company.

We have observed five broad categories of meta-information that can play a role in multiple repository use. All of these can be instrumental to system operation as well as directly valuable as an information resource for the user:

- *Content*—is information about what information is contained in the repository. This may be a textual description of the coverage of the database, a statistical analysis of the words used in the database, or possibly even a more structured representation of the content coverage using a knowledge representation language.
- *Source*—is information about the nature of the process that produced and/or maintains the repository. This includes information about whether the repository is, for example, a personal archive, a group dynabase, or a published public repository. Other important information includes the institutional source of the information, update frequency, mutability, and usage costs.
- *Form*—is information about the structure of information in the repository, in essence, typing information that defines the abstract layout of units of information. This kind of information includes a description of the

<sup>1</sup>Our design choice of search objects as single instances of search is motivated by the goal of concreteness. Arguably, search templates or other kinds of search class constructs may be workable alternatives or enhancements.

repository units in terms of their attribute structure, for example, what attributes each document has and the types of values stored for those attributes.

- *Functionality*—is the capabilities and properties of the retrieval engine that serves the repository. In particular, this information includes the kinds of searches supported by the engine, performance characteristics, repository location, and the nature of progress reporting supported. This information is primarily valuable for system operations, but can also be relevant to the user.
- *Usage Statistics*—is information about usage of the repository by individual users or by groups or categories of users. The primary use of this kind of information is to guide repository selection. For example, the user may remember using a particular source before or like to use sources considered productive by other members of a project team. This kind of meta-information raises some serious system issues regarding persistence and longevity of statistics and social issues regarding ownership, privacy, and access.

## 5 Repo

Based on the interaction model and repository modeling notions presented above, we have implemented a system called Repo that provides access to multiple, disparate repositories. As indicated above, our view of repository modeling requires that Repo support system operation and provide an end-user information resource. In particular, Repo currently represents the content, source, form, and functionality of a number of available repositories and their associated retrieval engines. Usage meta-information hasn't currently been addressed.

Our current implementation is written in Common Lisp (including CLOS) with a lightweight thread extension on Unix workstations. It is used as a library in the same address space as our current client, but it is conceptually separate and could be easily distributed using emerging object technologies.

The exported interface to Repo is mostly contained in six types of objects: registrar, repositories, search methods, search events, search results, and result items. The primary responsibilities of each these types is described below. (Other types of objects required by the interaction model, i.e. item stash, history, and visualizer are supported by a separate client library.)

**Registrar** A registrar plays two primary roles: first, it provides and maintains a registry of repositories, a meta-repository so to speak, including meta-information about each repository; second, it supports search on this repository, meta-search. To bootstrap, a user's information workspace

contacts a registrar, and uses direct lookup by name or meta-search to obtain one or more repository objects to search.

In addition to the registry of repositories, the registrar maintains a registry of docutypes, which capture meta-information about the structure of documents contained in registered repositories. In particular, a docutype specifies attributes of a document: this includes the names of the attributes, their types, and whether they are required and/or multivalued. Docutypes are organized in a single-inheritance hierarchy that allows sharing of attribute specifications among different docutypes. The docutype hierarchy is centralized so that it can be shared across different repositories.

**Repository Classes** Each registered repository is an instance of some repository class, that obeys a standard repository protocol, and possibly, additional specialized protocols. The two main categories of standard operations on repositories are document lookup and repository information lookup. In the first category, the main operations are listing document objects, and finding documents objects using repository-specific unique ids.

In the second category are operations that provide access to a variety of meta-information. This information is most conveniently associated with the repository class or instance, rather than being centralized with the registrar. In particular, it includes a textual description of the repository contents, a record of repository properties (i.e. source meta-information), the name of the root docutype of documents in the repository (i.e. form meta-information), and a list of supported search methods (i.e. search engine functionality meta-information).

Other useful operations provided on some repository classes include accessing word usage statistics for the repository as well as clustering services on collections of documents from the repository.

We have implemented a number of repository classes in the current system including the following:

- Protofoil—collections of scanned documents with text extracted using optical character recognition and indexed using TDB [7], a full-text search engine based on linguistic and statistical analysis of term usage.
- Dialog—online databases provided by Dialog accessed through a telnet connection on the Internet using Dialog retrieval capabilities. Dialog databases are variously organized and in many cases, support different retrieval capabilities. Further specialization of the Dialog repository class is necessary to model the different major types of Dialog databases.
- Dynabases—resource documents gathered and filed from email, news, and other available sources served by a combination of attribute/value search and TDB search methods. Additional protocol is provided for adding new documents to the dynabase.

- **References**—Published textual reference material (e.g. Grolier’s encyclopedia) served by direct name lookup and TDB search methods.

**Search Methods** Repository classes can be queried about supported search methods. Search methods describe search techniques supported by one or more repositories and characterize both generic properties of the method and specific details of composing a query for that method (e.g. data format). The generic properties allow the information workspace to select an appropriate user interface for searching the repository and visualizing search results, while the details allow the repository-specific queries to be formulated and specific parameters of the user interface to be filled in.

The generic properties characterize the properties of the technique and its implementation: these include the general kind of search (e.g. boolean expression on attribute values, regular expression on text, similarity using text or documents, or a proximity search i.e. for  $n$  particular words occurring within an  $m$  word window), the type of item returned (document, repository, set, or part), whether results are returned in ranked order, whether progress reports are supported, and whether results can be generated incrementally.

**Search Event** A search event is created by the information workspace to embody a new incident of search by the user. A search event provides operations for setting its scope, specifying its query expression, starting and managing a process to perform the search, reusing it to create further search events, and accessing its result. In addition, callbacks are supported on the client of the search event for notifications about progress and completion.

A search event’s query expression may consist of multiple search methods and its scope may contain multiple repositories, so Repo may have to perform complex query separations and result mergings. Though we have started building internal abstractions for managing this process and interactions on error conditions, this remains one of the primary areas where further work is required.

**Search Result** Search result classes define different kinds of structures for organizing the result of a search. For example, some search results may be complete, whereas others may only be partially enumerated at a given point. Appropriate visualizations can be provided by Repo clients based on the protocols supported by different result classes.

**Result Items** Result items are objects that are found by a search. Consistent with the interaction model, there are four types of result items: documents, document sets, document parts, and repositories. The actual class of a result item returned by executing any particular search method on any particular repository is determined by item type, search method, repository class, and possibly by docutype.

Thus result item objects support standard item protocols as well as additional protocols determined by the search method and the repository class. Document objects support operations for accessing their repository-specific unique id, their textual content (if any), their attribute/value description, and their docutype. In addition, the repository class may provide operations for accessing additional information for the item, e.g. image thumbnails, image views, or word usage statistics.

## 6 Labrador

A design for interaction between the user and a retrieval-enabled information workspace was presented above. That design is largely independent of detailed user interface design. In this section, we describe the user interface of the Labrador system based on the interaction model and the Repo repository access system. Labrador currently embodies most of the elements of the interaction model: further work needs to be done to support an item stash and spotlighting (i.e. highlighting the items that match a subquery within a visualization). Labrador retains many of the design elements that we developed in previous work on Infogrid[19] including a carefully managed spatial layout and gestural buttons, but extends the user interface vocabulary as needed by the interaction model.

**Layout** The Labrador screen is divided into tiled areas that are managed automatically and/or manually according to user actions and preferences. As shown in Figures 3, 4, and 5, the layout consists of three columns: a thin column on the left that contains gestural buttons that activate different browsers; a middle column that contains various panes for filling in a search request and visualizing search results; and a column on the right, which in Figure 4 contains a browser for visualizing specific items returned in the search and in Figure 5 contains a browser for visualizing the history of completed and active searches.

**Retrieval Loop** The user initiates a meta-search using the “Find Repo” tool (represented by the walking finger icons). This presents a search form, shown in Figure 3, that allows the user to fill in the name of the repository, repository type, required fields, or text that gets matched against the content description of the repository. The user can then start the search which will show matching repositories in a Search Result pane.

These repositories can then be selected as scopes, and a new search form based on those repositories can be created. Starting this search creates a new Search Result pane that contains item buttons that may represent specific documents, collections of documents, pieces of documents, or a mixture of items at these levels. Items can then be selected as “relevance feedback” to a pending search, as shown in

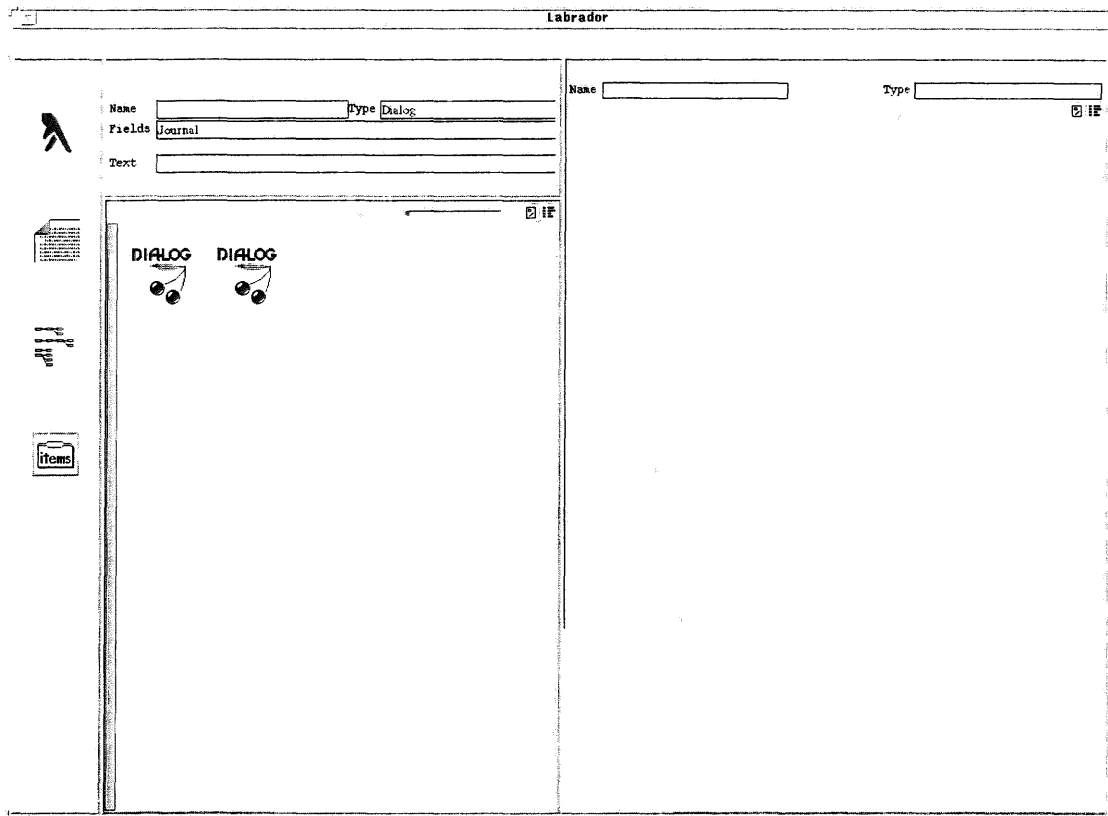


Figure 3: The user has performed a search on the Registry of Repositories for “Dialog” repositories that have documents with an attribute named “Journal.” The search has found 2 registered repositories.

the “Similar To:” pane of Figures 4 and 5, using a gesture. Gestures on the result header or set items will add contained documents to the scope of a new search, as shown in Figure 4.

**History Visualization** Creating new search forms adds nodes to the history visualization. Links indicate scoping. In particular, the result of the node at the link’s left endpoint is attached to the scope of the node at the link’s right endpoint. The strict tree layout means that the query is equivalent to a conjunction of all ancestor queries. Further work needs to be done to depict more complicated scoping patterns. Multiple inheritance lineage can be used for disjunction of queries, but this has implications for spatial persistence, layout, and automatic pruning. Matters are complicated further by scopes constructed with subsets of possibly more than one result.

The history visualization shows nodes in four possible states represented by different icons: awaiting initiation by the user, so still editable (open circle); stopped by the user or possibly automatically by the system (yield sign); currently running with a progress indicator (turning arrow); and finally finished (filled circle). The nodes also show the short name of repository searched, the number of hits, and possibly the name of the search as attached to the node by the user.

The user can pop-up a menu of operations by clicking on history nodes or use gestures: operations include pruning, naming, and attaching notes to nodes, continuing stopped processes, suspending or aborting running processes, and reusing underlying query specifications or results.

The history view can quickly get cluttered if all search objects are recorded. Requiring explicit pruning for all nodes puts too much burden on the user; thus we believe developing automatic history pruning policy is an important area for future work. Our initial policy design needs further experimentation and tuning as more complicated scoping patterns are supported. Nodes in certain states are retained: named nodes and their ancestors, and incomplete, stopped, or running nodes. All other nodes are pruned in least recently used order with a settable limit of descendants from the same scope. In addition, a much larger secondary buffer of pruned history nodes is managed. We believe that a two-level system can use valuable screen resource more effectively while still providing a reasonable safety net.

**Item Visualizations** Visualizer panes have visualization buttons in their title bar that switch between different visualizations of the pane contents. For example, the Search Result pane has buttons for visualizing the results in four ways:

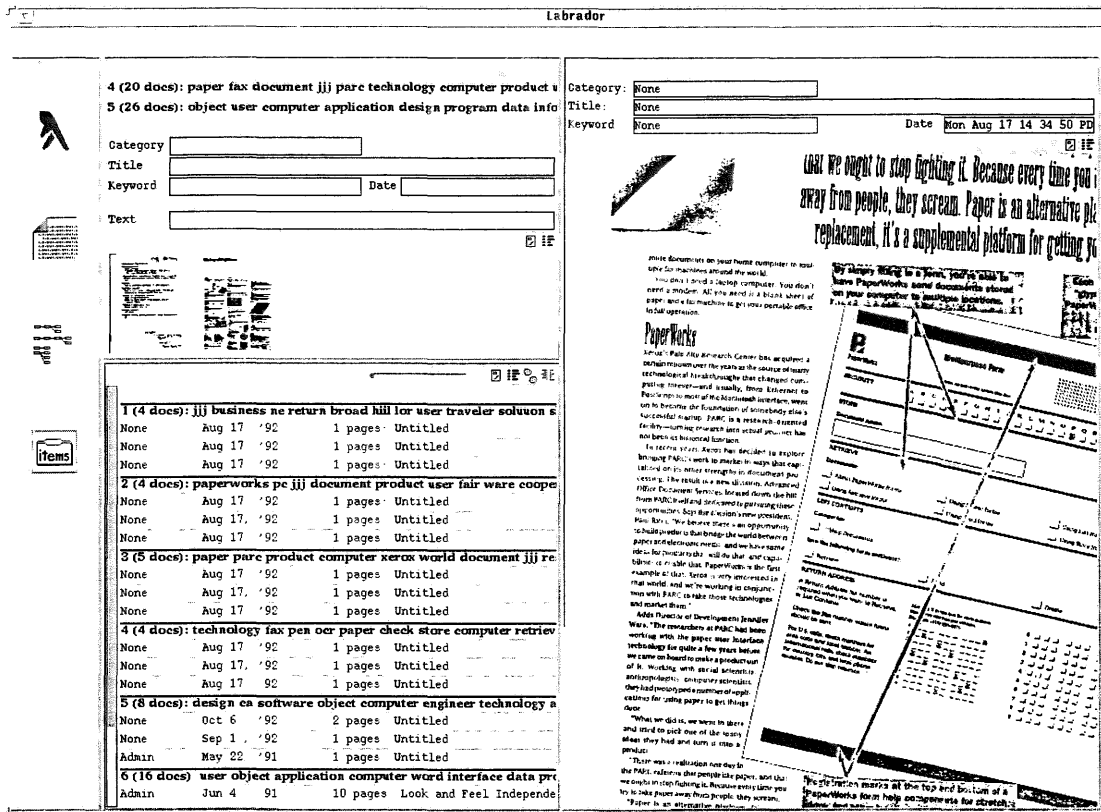


Figure 4: The user is viewing a collection of 46 scanned documents that were retrieved in a previous query using an automatic clustering view. The user browsed a document on the right side and is about to retrieve documents that are similar to a two relevant document.

1. an array of Infogrid-style thumbnails (shown in the Similar To: pane of Figure 4)
2. a list of description lines showing item attributes (shown in the Similar To: pane of Figure 5)
3. a list of automatically-generated clusters of documents with the central terms and documents for each cluster using techniques described in [6] (shown in Search Results pane of Figure 4)
4. a list of documents along with the snippets showing specific occurrences within each document using techniques described in [18] (shown in Search Results pane of Figure 5)

Document Browsers can support different renderings of the same document. The browser shown on the right side of Figure 4 supports two visualizations for scanned documents: one for the scanned pages and another for the OCR-ed text of the document.

## Conclusion

We have developed two key components that support the goal of building information access systems based on the

notion of embedded retrieval from multiple information sources. The interaction model is a design for organizing the interaction between the user and retrieval systems. The repository model is a design for capturing meta-information about repositories so that it may be used to integrate them into a common workspace as well as provide searchable, browsable information to the user about them. We have implemented these designs in a system called Repo and a user interface based on them in a system called Labrador. Further work is necessary to take full advantage of the power of repository modeling in the Labrador user interface.

Our experience in this project and others convinces us that the design of a usable and useful system depends on much more than the design of the user interface defined narrowly. It is our belief that *conceptualization* must precede *visualization*. Thus, we have gone to great length to understand the kind of interaction suitable to the nature of information gathering embedded in larger work processes, to develop an interaction model and the underlying concepts for supporting such interaction. Only then was this understanding captured in system components. This approach may be useful in other applications as well.

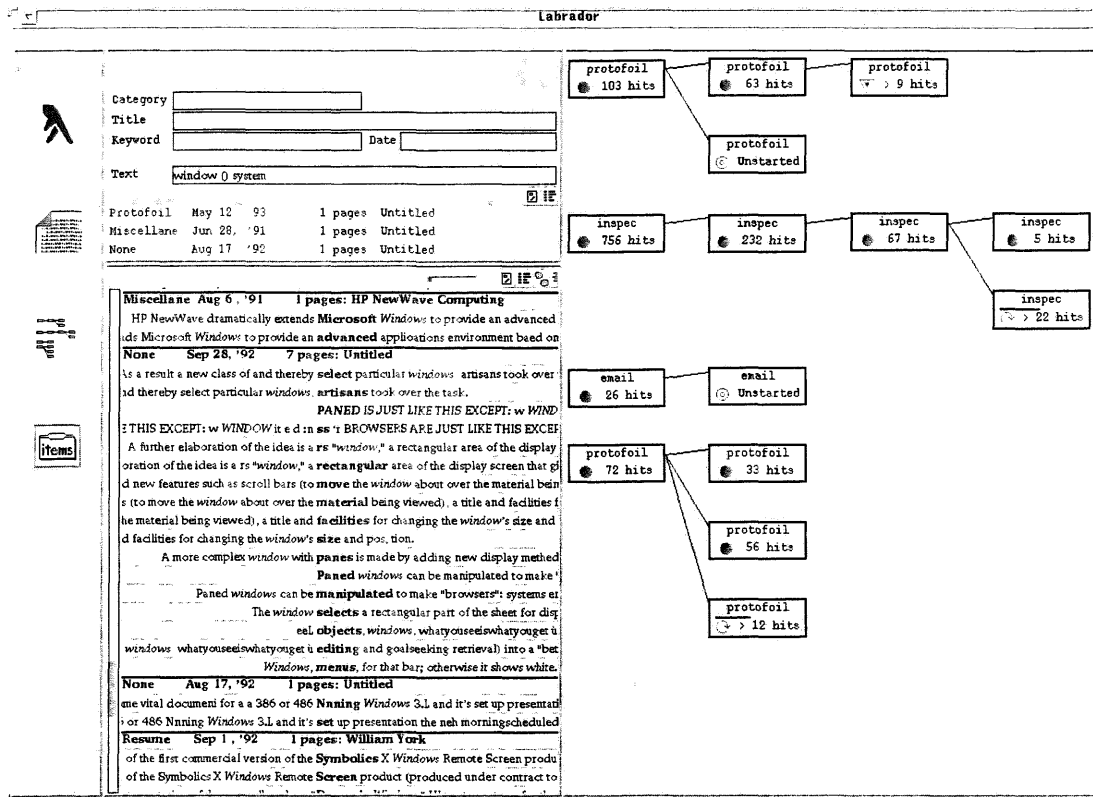


Figure 5: Documents containing “window system” have been retrieved and all occurrences within each document are being visualized as one line “snippets.” The history browser on the right shows the status of current and past searches and allows users to control, browse, and reuse underlying search processes and results.

## Acknowledgements

A number of our colleagues have contributed to this work in various ways. Richard Burton, Stuart Card, and Randy Trigg have contributed to ideas presented here as well as improved this paper. Doug Cutting, Jan Pedersen, and David Karger have provided us with a very powerful search engine and clustering techniques for dealing with text repositories. Larry Masinter, Steve Putz, and Bill Janssen have provided an illuminating information access infrastructure at PARC through their own work and the incorporation of Wais, Gopher, and World Wide Web.

## References

- [1] M.J. Bates. Where should the person stop and the information search interface start? *Information Processing and Management*, 26(5), 1990.
- [2] L.M. Berlin, R. Jeffries, V.L. O'Day, A. Paepcke, and C. Wharton. Where did you put it? issues in the design and use of a group memory. In *Proceedings of InterCHI '93*, 1993.
- [3] T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollerman. World-wide web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, 1992.
- [4] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 181–188. ACM, April 1991.
- [5] David Owen Case. Conceptual organization and retrieval of text by historians: The role of memory and metaphor. *Journal of the American Society for Information Sciences*, 42(9):657–668, 1991.
- [6] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR '92*, June 1992. Also available as Xerox PARC technical report SSL-92-02.
- [7] D.R. Cutting, J. Pedersen, and P-K. Halvorsen. An object-oriented architecture for text retrieval. In *Conference Proceedings of RIAO '91, Intelligent Text and Image Handling, Barcelona, Spain*, pages 285–298, April 1991.

- [8] Susan T. Dumais and Deborah G. Schmitt. Iterative searching in an online database. In *Proceedings of the Human Factors Society 26th Annual Meeting*, pages 398–402, 1991.
- [9] Thomas Erickson and Gitta Salomon. Designing a desktop information system: Observations and issues. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 49–54, 1991.
- [10] G. Fischer and H. Nieper-Lemke. Helgon: Extending the retrieval reformulation paradigm. In *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, pages 357–362, 1989.
- [11] D. Harman. *Relevance Feedback and Other Query Modification Techniques*, chapter 6. Prentice-Hall, 1992.
- [12] B. Kahle and A. Medlar. An information system for corporate users: Wide area information servers. *Online*, August 1991.
- [13] B. Kahle, H. Morris, J. Goldman, T. Erickson, and J. Curran. Interfaces for distributed systems of information servers. Technical report, Thinking Machines, Inc, 1992.
- [14] G. Marchionini. Interfaces for end-user information seeking. *Journal of the American Society for Information Science*, 43(2):156–163, March 1992.
- [15] M. McCahill. Ther internet gopher: A distributed server information system. *ConneXions — The Interoperability Report*, 6(7):10–14, 1992. Interop, Inc.
- [16] B.C. Neuman. Prospero: A tool for organizing internet resources. *Electronic Networking: Research, Applications, and Policy*, 2(1):30–37, 1992.
- [17] Andreas Paepcke. An object-oriented view onto public, heterogenous text databases. In *Proceedings of Data Engineering Conference*, 1993.
- [18] J. O. Pedersen, D. R. Cutting, and J. W. Tukey. Snippet search: a single phrase approach to text access. In *Proceedings of the 1991 Joint Statistical Meetings*. American Statistical Association, 1991. Also available as Xerox PARC technical report SSL-91-08.
- [19] R. Rao, S. K. Card, H.D. Jellinek, J. D. Mackinlay, and G. G. Robertson. The information grid: A framework for building information retrieval and retrieval-centered applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM Press, Nov 1992.
- [20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.
- [21] Dan M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. The cost structure of sensemaking. In *Proceedings of InterCHI '93*, 1993.
- [22] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, June 1990.
- [23] M. F. Schwartz, A. Emtage, B. Kahle, and B. C. Neuman. A comparison of internet resource discovery approaches. *Computing Systems*, pages 461–493, Fall 1992.
- [24] C. Stanfill and B. Kahle. Parallel free-text search on the connection machine. *Communications of the ACM*, 29(12):333–352, 1986.
- [25] M.D. Williams. What makes rabbit run? *International Journal of Man-Machine Studies*, 21:333–352, 1984.
- [26] M.D. Williams, F.N. Tou, R. Fikes., A. Henderson, and T.W. Malone. Rabbit: Cognitive science in interface design. In *Proceedings of 4th Annual Conference of the Cognitive Science Society*, pages 82–85, 1982.

