Integrated Information Retrieval in a Knowledge Worker Support System

Gordon McAlpine
FCI Informatics Research Centre
Buddinge Hovedgade 80, 2860 Søborg, Denmark.

Peter Ingwersen
Royal Danish School of Librarianship
Birketinget 6, 2300 København S, Denmark.

Abstract

This paper describes the design of the information retrieval facilities of an integrated information system called EUROMATH. EUROMATH is an example of a *Knowledge Worker Support System*: it has been designed specifically to support mathematicians in their research work. EUROMATH is required to provide uniform retrieval facilities for searching in a user's personal data, in a shared database of *structured* documents and in public, bibliographic databases. The design of information retrieval facilities that satisfy these and other requirements posed several interesting design issues regarding the integration of various retrieval techniques. As well as a uniform query language, designed to be highly usable by the target user group, the retrieval facilities provide *expert intermediary* functions, i.e. sophisticated support for the retrieval of bibliographic data. This support is achieved using a model of the user, a model of the user's information need and a set of search strategies based on those used by human intermediaries. The expert intermediary facilities include extensive help facilities, automatic query reformulation and browsing of a variety of sources of query terms.

1. Introduction

EUROMATH is an example of a type of system, which we call a **Knowledge Worker Support System**, that is likely to become more common as an application of information retrieval. It is designed to provide an *integrated* set of *support tools* for a *specific* target user group, who are characterised by the *tasks* they perform. The target users are all persons engaged in mathematical research. The target tasks for which tools are to be designed and implemented are the writing of mathematical documents, communication between mathematicians and the retrieval of information relevant to mathematical research. The main aim of the **EUROMATH Project**, which is being financed by the Commission of the European Communities, is to provide a *comprehensive* set of functions for the support of the target tasks in a way that is *easy to learn and use*. The system must also *integrate existing tools*, and corresponding data, with tools specifically implemented for the project.

The EUROMATH System has been designed to run on **high-performance workstations**, which have a large, bit-mapped screen, a powerful CPU and several megabytes of RAM. This has made it feasible to design an ambitious, advanced system using powerful user interface techniques. It is also assumed that, at each user site, a local area network connects workstations to various servers, including a file server on which a shared database can be stored, and a gateway to a wide area network, which facilitates electronic mail and access to public databases.

The **aim of our work** within the EUROMATH Project has been to design a common query language, query support facilities and browsing facilities that can be conveniently used to retrieve all the different kinds of data that will be available to mathematicians in the EUROMATH System. Our work has had a practical, concrete goal and was not aimed at tackling any specific research issue. In the first half of this paper (section 2), we describe the retrieval functions and user interface that we have designed to access both **internal data** generated within EUROMATH and **external data**, e.g. public databases. This description relates our design to the research ideas and state of the art techniques that we have attempted to apply. The second half of the paper (section 3) describes the additional expert intermediary facilities that we have designed to support users in retrieving bibliographic data from external sources. Our work in this area has taken a rather novel approach, which concentrates on the use of complex search strategies, based on the techniques used by human intermediaries.

2. Retrieval of Internal Data

The requirements resulting from our aim of providing **comprehensive functionality** that are most relevant to the work described in this paper are:

- **powerful document processing** facilities based upon (hierarchically) structured documents, as further described in [Draper 88], [Horak 84] and [Furuta 88].
- facilities for the filing and retrieval of data from other systems (i.e. **external data**).

The most relevant requirement resulting from our aim of **ease of use** is the need for tight **integration**, i.e. for the integration of both the data and the functions of all the tools to be provided within the system, hereunder the external, bibliographic databases.

Like so many users of computer technology, a mathematician who wishes to make comprehensive use of computer-based support at the present time is faced with a jungle of products and services, which are often incompatible, and almost never integrated. In particular, regarding information retrieval, such a user will typically have to learn the different retrieval facilities of an operating system, a text editor, a mailing system, each database system that provides relevant information, etc. In addition, such a user may have many other kinds of information needs for which powerful retrieval facilities would be useful, but are not provided, e.g. searching information on how to use the system. A major goal of the EUROMATH Project is to provide a single, *uniform* set of retrieval facilities that can be used for all the tasks of a mathematician.

We decided that this combination of requirements could best be fulfilled by basing EUROMATH on an **object-oriented approach** [Wegner 87] to both the system design (data modelling) and the user interface. All data are represented as objects. An **object** is some meaningful collection of data that represents an entity from the application domain, e.g. a mathematical paper or a formula. Each object is represented as a set of attribute values, which may be simple values, subordinate objects (called **components**) or references to other objects. Each object belongs to a **class**, which defines both the attributes and the set of operations that may be applied to objects of that class. The classes of objects are organised in an inheritance hierarchy, so that common operations are shared amongst all objects to which they apply.

As in the Xerox Star [Smith 82] and the Apple Macintosh, the user interface is based on the principle of **direct manipulation**: the system displays objects on the screen, e.g. as icons or windows, and the user manipulates these objects by using a mouse to firstly select the object of interest and thereafter select the required operation from a pull-down menu. But the EUROMATH System is more tightly integrated, since whenever an object is selected, the system knows which operations are applicable to objects of that class, and precisely those operations will be made available through the system's menus. Thus, a formula or drawing within a mathematical paper may be directly modified in the window displaying the paper. Further, the system is highly **orthogonal** (or modular), since an operation that can be applied to one particular object in one particular situation becomes available for all objects and situations in which it is applicable. So, for example, the mailing tool does not need to have its own retrieval facilities: the general retrieval operations can be used for retrieving messages, just like any other objects.

2.1. Filing and Browsing

Our approach also facilitates more precise retrieval. The classification of objects into classes allows a user to narrow the search space, if he knows the class of the object he is looking for. The explicit representation of the properties of objects as a set of attributes allows more accurate querying than if objects were just represented as unstructured text in operating system files. The querying facilities that are based on this kind of filing are described in the next subsection. They are primarily suitable for retrieval from large data collections, which are typically shared amongst many users, and whose contents may be unfamiliar to the individual user.

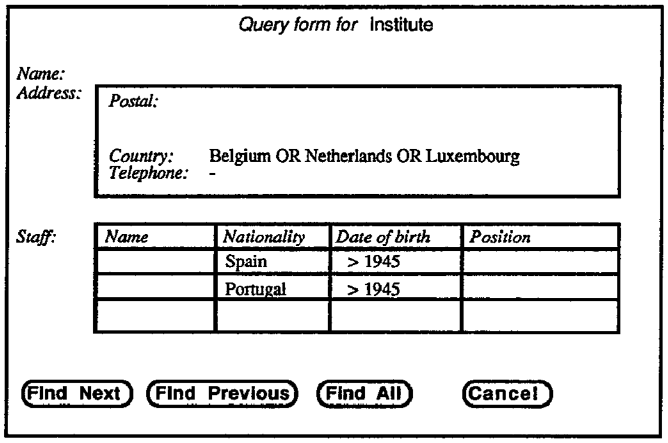
For retrieving more familiar data, typically created by the user himself, EUROMATH will provide **browsing** facilities based on **folders** that are similar to those of the Macintosh. A user can place each object in one or more folders. Each folder may be placed within another folder, forming a user-defined classification hierarchy. The browsing facilities for folders facilitate retrieval by means of *perceptual cues*: each object within a folder will by default be presented as an *icon* (a small, characteristic picture), which the user can place at a suitable position within the folder's window.

To increase the usefulness of the folder classification, we have integrated it with the querying facilities as follows. The set of folders in which a particular document is placed is represented as an attribute of the document. This means that folders also function as a kind of keyword facility, since search criteria can also be specified for this attribute.

The remaining **browsing facilities** of EUROMATH are primarily made up of a set of flexible information display facilities, which are designed to make the information relevant for browsing easily and quickly comprehensible to a user. Whenever possible, this is achieved by means of **perceptual encoding**, in which the relationships between objects that are known to the system are presented graphically. For example, the folder classification hierarchy is displayed as an acyclic, directed graph, which is illustrated in use for another purpose in subsection 2.3. Browsing is also supported by the fact that, as in **hypertext** systems, a reference to an object displayed on the screen as part of another object can be selected and the corresponding object displayed in a separate window. The browsing facilities are more fully described in [Draper 88], and some general principles for the graphical presentation of data that are useful in the design of browsing facilities are described in [McAlpine 88].

2.2. Query by Forms

To satisfy the requirement for ease of use, we have chosen to base the query language on the concept of **query by forms** [Zloof 81]. Basically, a query form consists of a set of fields - one for each attribute. Each field consists of a fixed header and a value specification. An example of a form for searching in the Institute class is illustrated below. The corresponding query is defined by ANDing together the criteria specified in each field. Within each field, values from the attribute's domain may be combined by the AND, OR and NOT operators. Further, each value may be preceded by a relational operator, e.g. ">", which is to be applied to the values of the searched documents.


Query form for Institute

Name:
Address:
Postal:
Country: Belgium OR Netherlands OR Luxembourg
Telephone: -

Staff:

Name	*Nationality*	*Date of birth*	*Position*
	Spain	> 1945	
	Portugal	> 1945	

Previous work, e.g. [Greenblatt 78], suggests that a **forms-based** query language allowing simple combinations of criteria composed of constant values is easier to learn and use than a command-line-based query language. A user does not need to remember attribute names and, in EUROMATH, filling in a query form is almost identical to the creation of a new object - the user can type strings and numbers, he can select values from option menus and he can paste both values copied from other objects and references to other objects (as in hypertext systems). Further, a user may convert an object to a query by simply copying that object and then pasting it onto a query form. This results in the copying of the values of all the attributes the object has in common with the query form. Then the user need only modify those query criteria that differ from the previously found object. This facility supports *iterative* querying based on feedback from the retrieved objects, which is particularly useful when users have difficulty in formulating a vague information need as a query.

In EUROMATH, the value of an attribute may also be either another *object*, which is called a **component**, or a **reference** to another object, which represents a *relationship* between two independent objects. The field value specification for such attributes is itself an **embedded form**, e.g. the *Address* attribute above. Since a component is also an object, it may itself contain components, and in this way, **hierarchically structured objects** can be built.

The value of an attribute may also be a *set*. The field value specification for such attributes is a **table**. The criteria specified within each row of a table are ANDed together, and each of the resulting queries is in turn applied to the *set* of values of the corresponding attribute, i.e. retrieved documents must have a set of values for this attribute such that the query specified in each row of the table is fulfilled by at least one member of this set. For example, the query form illustrated above would retrieve all mathematical institutes in one of the Benelux countries that have both Portuguese and Spanish members of staff who were born after 1945.

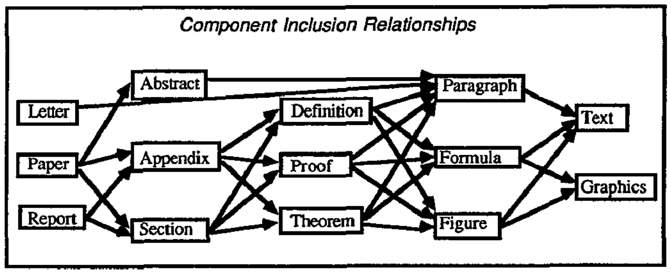
Components, references and sets are directly represented in our **underlying data model**. As well as providing powerful support for the EUROMATH tools, we believe that this representation is closer to the way that users conceptualise information, and thus simplifies the process of query formulation for the user, [McAlpine 85] and [Harper 85].

A single query form cannot express certain kinds of **complex information needs** that can be expressed in the boolean command languages of external database hosts, e.g. a predicate on one attribute ORed with a predicate on another attribute. However, such complex information needs can be expressed in EUROMATH by combining the results of several queries using the standard **set operations**: union, intersection and difference, which are provided as general commands.

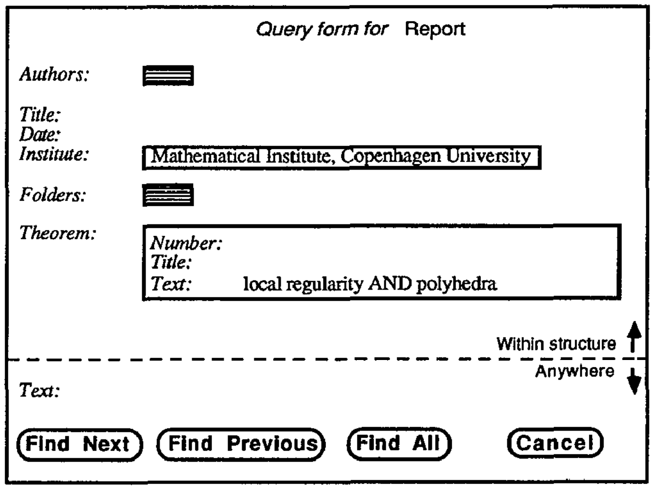
2.3. Hierarchically Structured Objects

The query form illustrated above showed how the query language can deal with **hierarchically structured objects** by means of nested forms and tables. Other objects in EUROMATH, especially documents, will tend to have rather *complex* hierarchical structures, as described in [Draper 88] and [Horak 84]. For example, a mathematical Paper might include an Abstract, a set of Sections and a set of Appendices. Each Section might include Definitions, Proofs and Theorems. Each of these may include Paragraphs, Figures and Formulae, which in turn might be composed of basic Text and Graphics components. Suppose a user wishes to retrieve a report he has previously read from the shared, internal database, and he knows that it originates from Copenhagen University, and that it includes a theorem about local regularity of polyhedra. Then, using query forms as described above, the user would need to know and specify whether the theorem was in the main sections or the appendices, and the form would include all the attributes of the report, about which the user knows nothing.

To avoid these problems, we provide a more **flexible solution**, based on the ideas of Barbic and Rabitti [Barbic 85], in which a user can build up an appropriate query form with those components about which he can supply some information. When a user invokes the Search operation for a class of objects that have a complex structure, the system will display both a query form and a component selection window. The query form will, by default, only display a few of the most useful first-level attributes of the selected object class, e.g. the author, title, date and institute of a report. The **component selection window** displays the names of all the component types that can be used to compose objects of the specified class. The user can select one of three presentation forms for this window: an alphabetically-ordered list, an acyclic, directed graph showing *inheritance* relationships, or an acyclic, directed graph showing *inclusion* relationships, as illustrated below. The latter two are examples of the technique of *perceptual encoding* mentioned earlier.



From any of these windows, a user can copy the components about which he can specify information to the query form. The system will automatically arrange the components in the query form to reflect the inclusion relationships of the class definition, e.g. Text will be nested within Theorem, if both are copied, as illustrated in the diagram below. But note that the levels of the structure corresponding to those components that have not been copied by the user, e.g. Paragraph, are missing from the form. A user may then enter the information he knows.


Query form for Report

Authors:

Title:
Date:
Institute: Mathematical Institute, Copenhagen University

Folders:

Theorem:
Number:
Title:
Text: local regularity AND polyhedra

Within structure
Anywhere

Text:

In general, this facility is **useful in situations** in which a user knows that the information he requires is represented in a particular way, i.e. as a specific component within a complexly structured object. In EUROMATH, it is primarily intended for searching in collections of complexly structured objects that a user has previously read, i.e. personal collections of papers concerning the user's research interests. In other Knowledge Worker Support Systems, it might also be used for searching within collections of complexly structured objects in which certain types of information are represented in a uniform way, e.g. the conditions for qualifying for a particular kind of social security benefit in a database of social laws and regulations.

Traditional **text-matching queries** are handled by letting all the text within an object belong to a built-in class called Text, which is treated just like any other component class. In other words, the user can select *Text* from the component selection window if he wants to specify that retrieved documents contain a given text string, as illustrated in the example above. By default, the Text component is placed within any other selected component that can include Text, e.g. Theorem in the example above. In this case, only reports for which the given text string occurs within a Theorem are retrieved. However, users will typically not know within which component the text they are looking for occurs. To cater for this, a user can specify that a component may occur **anywhere** within the document structure. This is done by moving that component into the specially marked part of a query form below the dashed line, which is also illustrated in the Report form above. A similar idea has previously been suggested for text-valued attributes only [Bertino 88]. We believe that it is useful to extend this to allow a user to specify that any component may occur anywhere within the structure.

A consequence of our goal of providing a single uniform set of retrieval facilities is that they must also be used for searching ***within*** **a single document** - usually a dedicated function of a text editor. This is facilitated by introducing the concept of a scope. The **scope** of a query is a set of one or more objects, typically folders, to which the search is to be restricted. When the cursor is positioned within an object being edited in a window, and the Search operation is invoked, then the scope of the search is implicitly defined to be that object. The search scope may also be explicitly defined by selecting a part of an object, a set of objects, or an entire class of objects.

For text and string valued attributes, **special symbols** (?, # and !) are provided for truncation and masking. This caused an *integration problem*, since in the external databases, truncation may only be used at the end of a word, while this restriction is not necessary when searching within a single object, since it is not implemented with inverted files. We have chosen to also only allow truncation at the end of words when searching within a single object - for the sake of uniformity.

2.4. Discussion

The basic EUROMATH retrieval facilities, which have been described in this section, must fulfill a wide variety of requirements that cover all retrieval situations that can occur in the various EUROMATH tools, including complexly structured objects, uniform access to both external and internal data, support for both factual and topic queries. The resulting query language bears greatest similarity to the forms-based query languages that are present in many fourth generation tools for Relational Database Management Systems, which have their roots in Zloof's Query-By-Example. However, the EUROMATH retrieval facilities differ in several ways: they can handle more complexly structured documents, they facilitate advanced browsing, they provide more powerful text retrieval, they do not support highly complex fact queries and they provide extra support for retrieving bibliographic data, as described in the next section.

The design of the query language for a Knowledge Worker Support System such as EUROMATH has to find a suitable balance between the requirements for comprehensive functionality and ease of use. But what is sufficiently comprehensive functionality?

One approach is to provide a query language that is "complete" in the sense that a relational algebra is complete. However, this kind of **completeness** is only measured relative to the mathematical model on which the language is based. For example, in practice it has been shown that problems occur in using relational databases because they are not complete with respect to the application domain: e.g. loss of semantic integrity due to the lack of an entity concept [Schmid 75], and the lack of constructs for recursive queries [Heiler 85].

We have therefore taken the approach of evaluating the querying constructs described in the literature on object-oriented and semantic databases relative to our understanding of the requirements of mathematicians. We found no descriptions of experiences in using advanced systems like EUROMATH that could help us make these decisions. We suggest that **empirical research** into the use of such systems in practice is of crucial importance to the future design of similar systems and the identification of related research issues.

We have also investigated whether our design could be extended to handle a much more comprehensive functionality, including such constructs as the existential quantifier, aggregate functions, query variables (as in Query-By-Example) and a subset operator for set-valued attributes. Most of these extensions are related to supporting the kind of **complex fact queries** which are commonly used in database management systems, e.g. how many Russian mathematicians have never written a paper in English? Although we found that it was quite feasible to extend our design to handle these constructs, we have not included them because these kinds of complex fact queries will rarely be of use in the tasks that the EUROMATH System is designed to support. This decision might have been different if our target domain had, for example, been an office information system including integrated administrative data processing.

Similarly, we have considered extending our design to allow queries to be specified in terms of the **layout structure** [Horak 84] of objects, e.g. specifying a piece of text to be retrieved in relation to some "landmark", such as a figure. Although we found that it was also quite easy to extend our approach to cater for these kinds of queries, we decided that they are not necessary when powerful facilities exist for searching objects based on their contents.

3. Support for Bibliographic Data

According to the results of surveys and interviews carried out within the EUROMATH Project, very few mathematicians use the comprehensive, online databases of mathematical literature that are available. The two main reasons seem to be the relatively high costs and the difficulties of using the system. However, the hosts' **pricing schemes** for the databases of most relevance to mathematicians have recently been changed to encourage more usage, and CD-ROMs are also beginning to become available with bibliographic databases for mathematics. The **usage difficulties** are both due to the inherent characteristics of shared, textual data and to user interface problems. For example, users may not know the exact structure of the shared data and the conventions that have been used when assigning values to certain attributes. Retrieval from large *text* databases is a problem in itself, due to the multitude of possible ways of expressing the same information as text. Today, most of the difficulties of using the available host systems can be overcome by using skilled human intermediaries. However, this solution is often inconvenient, especially if mathematicians are to make more frequent use of these databases. Our goal within EUROMATH is therefore to provide a means whereby mathematicians can themselves access bibliographic databases. This section describes the expert intermediary facilities that we have designed to fulfill this goal.

Although it has been shown that a *partial match* (e.g. probabilistic) approach provides more effective retrieval [van Rijsbergen 86], we have chosen to base the query language, which was described in the previous section, on the *exact match* approach of **boolean logic.** The underlying reasons for this choice are that it is suitable for supporting the requirements for fact queries and retrieval of complexly structured documents described in the previous section, and that boolean logic is what is provided by the important external databases. Regarding the second reason, previous work [Morrissey 82] has shown that it is *possible* to implement probabilistic methods on top of boolean query languages. To our knowledge, however, these techniques have never been proven in a commercial environment, and would seem to give rather poor response times. In addition, they do not provide as substantial gains in retrieval effectiveness as other more sophisticated probabilistic methods. Considering that our target user group (mathematicians) do not seem to have the usual difficulties in understanding boolean logic, we decided that these disadvantages outweighed the advantages of a probabilistic-based front-end. However, if resources are available, we later propose to extend the system with probabilistic-based relevance feedback, e.g. as described in [Morrissey 82].

Having decided not to attempt to hide the weaknesses of the underlying systems, we have taken the approach that the users must become aware of them, and, with the help of the system, learn some of the **techniques** that can counter them - just as **human intermediaries** do. Many of these techniques can in fact be thought of as pragmatic ways of getting round the weaknesses of boolean logic by reinterpreting users queries. Some of these techniques will be built into the EUROMATH expert intermediary facilities, while we hope that the target user group will gradually be able to learn the most critical of the remaining techniques. Evidence to support this assumption came from a set of field studies performed within the project, in which the use of the external databases by mathematicians was observed, and from previous reports on the training of end-users [Haines 82].

In contrast to many previous expert intermediary systems, we have not tried to model **domain knowledge**. Instead, we leave the domain knowledge in the hands (or head) of the user, and concentrate on supporting him in interacting with the system and teaching how to translate his domain knowledge to the system's representation. We also try to exploit some of the more sophisticated facilities of the host systems, e.g. frequency analyses, with the aim of triggering the user's domain knowledge. Thus, our system is rather domain independent, but only suitable for users who are experts in a limited domain, covered by a few databases, who are willing and capable of quickly learning online searching.

Our system provides a fairly **complex set of search strategies**. These have been based on one of the authors comprehensive knowledge of the strategies of human intermediaries. Resources were not available for a more extensive form of knowledge acquisition, such as interviewing a representative set of human intermediaries. Using these strategies, the system can, when appropriate, automatically reformulate queries and supply a user with a set of alternative results that may be closer to his true information need than his possibly inaccurate expression of that need as a query. Of course, a computer-based intermediary has certain disadvantages in comparison with a human intermediary. However, within its limitations, its search strategies can be more systematic and comprehensive than a human intermediary, since it can retrieve a large number of complex sets quickly, it can keep track of a large number of alternative strategies and it can pursue all relevant possibilities.

The EUROMATH expert intermediary consists of the following **facilities**.

- Access to appropriate data that may be **browsed** in order to find suitable terms for queries.
- The **sequence of windows** shown to the user and their exact content will vary according to the user's characteristics and information need.
- There is a comprehensive, context-sensitive **online help** document available to users, on their own initiative, at all times.
- Under certain circumstances inferred by the system, appropriate parts of the online help document, and other relevant data are **automatically** shown.
- Under certain circumstances inferred by the system, the query specified by the user is **reformulated** to retrieve alternative result sets, which are presented to the user.

The basis for these facilities is a stereotype classification of a user, a description of his information need (in addition to his query) and a set of rules for inferring situations in which active help or automatic reformulation may be appropriate.

3.1. Modelling of Users

Users are modelled simply by dividing them into the following **user categories:**

a) **novice** or casual users, i.e. searchers not familiar with online information retrieval, query languages, search strategies, etc;

b) **semi-experts**, i.e. searchers with some experience of using query languages, but without detailed knowledge of the different host languages and complex search strategies;

c) **full experts**, i.e. searchers who have detailed knowledge of the different host query languages and complex search strategies.

These categories can be thought of as defining 3 different **modes of interaction** between a user and the system. In **novice mode**, the system will take the user through a sequence of windows, some of which are skipped in the two expert modes. However, in general, all of the facilities are available in all modes. The difference is that in novice mode, the user is *forced* through certain steps and the system will more frequently *automatically* intervene. The content and appearance of particular types of windows will also be tailored to the system's models of the user and his information need. The described design makes it rather clear to novice users when the system is intervening, and thus it should be easy for a user to change over to the expert levels if he uses the system a lot. In **semi-expert mode,** the same forms-based query language is used, but a user is informed of certain extensions and the system does not automatically intervene so frequently. In **full expert mode**, queries are specified in the host query language, i.e. the user can have a transparent dialogue with a host.

Users may themselves define which category they belong to by selecting the appropriate category from the *Experience* field of an *information need form*, as illustrated below. The remainder of this paper concentrates on describing the facilities provided for novice users.

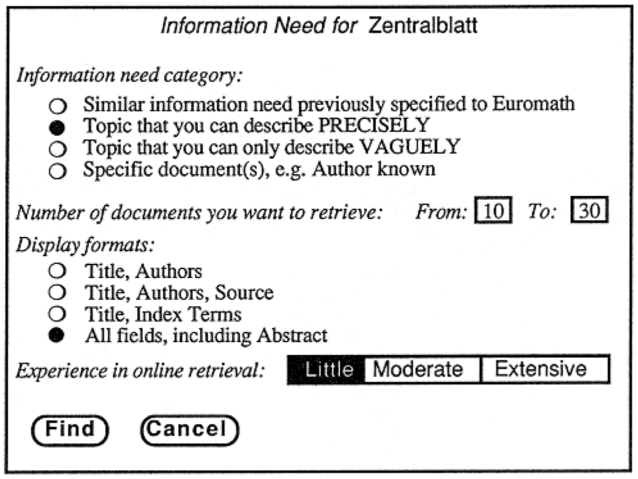
3.2. Modelling of Information Needs

A standard icon within each user's desktop will be a folder containing all shared databases both internal and external. *As for any other data*, a user can create a query form by selecting the appropriate database (i.e. class) followed by the Search command. *Automatic* **database selection** is not necessary in EUROMATH because there are only about 12 shared databases within EUROMATH. When a novice user has invoked the Search operation for a shared database, the *information need form* illustrated below will be displayed instead of a query form. The primary purpose of this form is to supply the system with information about a user's information need. In addition, it acts as a means for the user to structure his information need in his own mind.

Previous research [Ingwersen 86] has shown that three different types of information needs (or problems) can be distinguished:

1. **Verificative** needs (called "Specific documents" in the form): when a user searches for a single, well-known document or a set of documents by the same author. Users know certain specific data about the document, and want to retrieve it in full detail.
2. Well-defined, **topic** needs (called "PRECISELY" in the form): which are conscious problems, for which users know about the gap of information they require (what they don't know) and are able to specify it as a set of concepts.
3. **Ill-defined,** topic needs (called "VAGUELY" in the form): where the users do not have specific ideas about what they don't know. They only have a few, rather broad concepts in mind. Thus, they will require immediate conceptual feedback from the system, e.g. automatic support for browsing.

The **information need category** option of the information need form attempts to accommodate these 3 types of needs, as well as explicitly supporting situations where the user has searched for a similar information need previously.


Information Need for Zentralblatt

Information need category:
- ○ Similar information need previously specified to Euromath
- ● Topic that you can describe PRECISELY
- ○ Topic that you can only describe VAGUELY
- ○ Specific document(s), e.g. Author known

Number of documents you want to retrieve: *From:* 10 *To:* 30

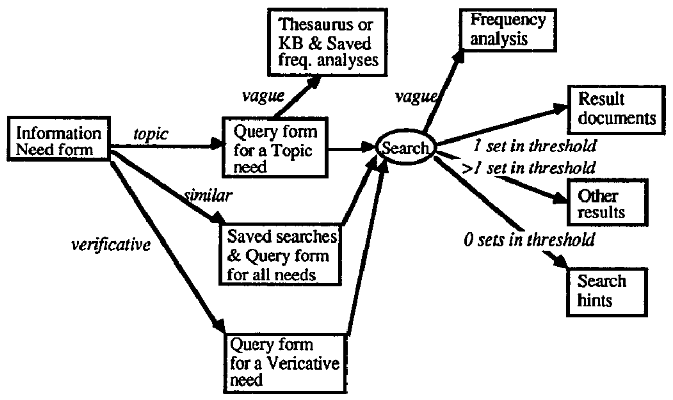
Display formats:
- ○ Title, Authors
- ○ Title, Authors, Source
- ○ Title, Index Terms
- ● All fields, including Abstract

Experience in online retrieval: [✓ Little] Moderate Extensive

Users can further characterise their information need by specifying what they expect to get back from the system. This is done by simply entering the minimum and maximum number of documents that the user wants to retrieve. The system can compare what the user specifies for these **threshold** values with the selection of an information need category. For example, if the specified need is verificative, but the lower threshold is greater than 5, then the system will later, if certain other conditions are fulfilled, ask the user if either the category or threshold is incorrect.

The **sequence of windows** shown to a novice user after the Find command has been selected from the information need form will depend on the information need category selected and the result of the search with respect to the specified threshold, as illustrated in the diagram below. If a user selects **verificative need**, then the system displays a query form suitable for verificative searching, with fields for author name, title, publication date and document type. While filling in the form, users have access to all the support facilities described in the next subsection. After a user has filled in this form, the system searches the external database using a search strategy of a similar kind to that described in the next subsection, i.e. which retrieves, and sometimes displays result sets other than the result set explicitly specified by the query. The most important part of the strategy, which is described in detail in [Draper 88], is the distinction of the following three cases:

1. If a user either fills in the author field or provides sufficient criteria on the other attributes (e.g. 2 title terms and the source), then direct online searching takes place. In the latter case, the user presumably does not remember the author, but *is* searching for a *single* document about which he can specify some facts.
2. If a user neither specifies the author nor sufficient alternative data *and* the specified upper threshold is less than 5, then the system will explain to the user that he must either specify more criteria or modify the category or threshold in the information need form.
3. Otherwise (i.e. as for (2), except that the upper threshold is at least 5), the system replaces the verificative need form with a topic need form, filled in with the attributes the user has already supplied, and thereafter functions as though the user had specified a well-defined topic need.

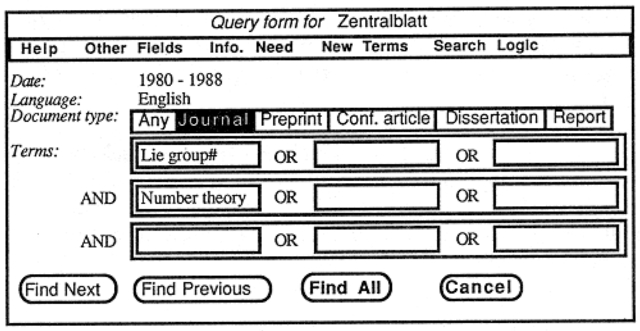


If a user selects **similar information need**, then the system will both display a query form suitable for all information needs (i.e. with all the fields for both verificative and topic needs) and a saved search index. The function of a **saved search index** is to allow a user to retrieve data saved during previous retrieval sessions and copy terms from them to the new query form. It contains entries for queries, frequency analyses, expand lists, sessions and results (all of which are described in the next subsection). Each of these entries is in turn an index, which can be opened and displayed as a table. To aid users in retrieving these saved data, the system will automatically assign index terms to a *Keywords* attribute of the saved objects. For example, for queries, the keywords will be all the terms entered in the query form.

The references stored as part of the saved results will be stored in a manner that facilitates integration. Each saved bibliographic record will be associated with a unique identifier consisting of the name of the database, the record number (accession number) and the update code (date of last change). This information will make it possible to combine the sets of records from various sessions in ***internal* bibliographic databases** which do not contain duplicates. Further, these internal databases can be searched in exactly the same way as the external databases, using the same query forms. Thus, a user can reuse an internal query on an external database, or browse the internal database to find terms for an external query.
The unique identifier of an external document reference is also used in the integration with the rest of the system: e.g. when a user copies a reference to a document into an e-mail message, it is stored as this unique identifier and displayed as a hypertext box, to indicate that it may be displayed in a separate window.

3.3. Searching for Topics

If a user selects either a precise or vague **topic need**, then the system displays a query form suitable for topic searching, as illustrated below. The *Terms* field corresponds to the basic index, except for certain cases described later. Expert users may also add field codes, e.g. /AU or /CC to the terms in this field. For novice users, **text-valued attributes**, e.g. *Terms*, are displayed as a matrix of term boxes. A user can enter a phrase in each term box. The boxes in each row are ORed together, while the resulting row criteria are ANDed together. The reason for using this matrix is to provide the user with a hint on how to structure the combination of criteria on text-valued attributes. The implied structure is a separate set of concepts, all of which should occur in the retrieved documents, and which are therefore ANDed together. Each concept can be expressed in different ways, and is therefore specified by a set of alternative phrases, which are therefore ORed together. A consequence of this design is that novice users cannot use the NOT operator. This is a deliberate decision: the NOT operator is hard to understand and rarely necessary.


Query form for Zentralblatt

Help Other Fields Info. Need New Terms Search Logic

Date: 1980 - 1988
Language: English
Document type: Any [✓ Journal] Preprint Conf. article Dissertation Report

Terms:	Lie group#	OR		OR	
AND	Number theory	OR		OR	
AND		OR		OR	

Find Next | Find Previous | Find All | Cancel

The first menu available along the top of a query form allows the user to display a window showing an appropriate part of a structured help document. The content of the help document is tailored to the user category, e.g. expert users will require more elaborate explanations of Fields and Codes and Display Formats and explanations may include concepts like basic index. Help documents are hierarchically structured and have hypertext properties, i.e. whenever a topic described in more detail elsewhere appears in a window, it is surrounded by a box to illustrate that it can be opened. In accordance with our principle of orthogonality, the help document may also be searched just like any other object. The **help menu** provides five different categories of help, including hints on what to do next, an explanation of the current state and an explanation of the currently selected item.

As for all query forms, the **Other Fields** command is available if the user wishes to specify criteria for attributes other than those displayed. The **Info. Need** command allows a user to modify the specification of his information need at any time. The **Search Logic** command allows users to inspect the sequence of search commands that are sent to the host as a result of the search strategy. Expert users may also modify these commands.

In order to support querying as an *iterative* process, in which a user repeatedly reformulates his query to more accurately reflect his information need, EUROMATH both facilitates users in copying terms from the documents displayed in the previous results (as described in the beginning of subsection 2.2) and provides the New Terms menu. The **New Terms** menu gives access to objects that may help the user find new terms to narrow or broaden his query, i.e. expand lists, frequency analyses, saved searches and a thesaurus or domain knowledge base.

Expand lists are alphabetically ordered lists of values that a particular attribute may take. The display of such a list is focussed around the term in the query form that the user has selected before invoking the Expand List command from the New Terms menu.

A **frequency analysis** lists, in order of decreasing frequency, the terms that occur in one or more specified fields of the retrieved documents. It is generated using the Zoom command on ESA or Select on STN. These commands provide some of the information that is automatically used in certain statistically based retrieval methods, and since their introduction in several public database hosts, have been found extremely useful by human intermediaries [Ingwersen 88].

The **thesaurus** that is available from the hosts is the Mathematics Classification Scheme, which represents 3 levels of narrower/broader term relationships. In addition, if resources permit, we intend to provide a knowledge base, in the form of a *term relationship network* [Larsen 87], which can cover more terms and relationships, which can be tailored by the individual user to his area of interest, and which can also be used in automatic query reformulation, as discussed later.

Returning to the query form: if a user has selected a **vague information need**, then, as soon as the user has entered the first term, the system will **automatically display** a thesaurus, any saved frequency analyses that are indexed by the specified term and a window informing the user that related terms from either of these windows may be copied to the query form. Regardless of whether the user actually copies terms or not, selecting Find All will cause the system to go online and perform the corresponding search, and replace the displayed saved frequency analysis with a new one obtained from the host. If the user is not satisfied with the first search result, then he may copy terms from the frequency analysis, or any of the retrieved documents, into the query form and repeat the search. (This can be likened to a kind of manual relevance feedback.) If the search algorithm, as described below, still retrieves a number of documents greater than the upper threshold, then the cycle is repeated once more. If the result is still above the upper threshold, then a specially tailored search hints window, which explains the use of more specific terms and other fields, is displayed instead of a frequency analysis. Thereafter, the system functions as for a *precise* topic search.

Users will often find it difficult to correctly categorise their information need. A general problem is that many users with well-defined topic needs often formulate their problems as a *label*, i.e. in the rather imprecise terms with which they briefly characterise the problem for themselves [Ingwersen 88]. Thus, a user who has chosen the *precise* need option may actually supply information in the query form that is more characteristic of a *vague* need. In such a situation, **the system infers** that the user actually has a *vague* information need and responds correspondingly, as described above. This situation is inferred if a user chooses the precise topic need option and only fills in one or two search terms, and the number of documents retrieved is higher than the specified upper threshold.

The **basic search strategy** used by the system to compose a sequence of commands to be sent to the host is given below. Thereafter it is also illustrated by means of an example. The basic idea is that if the result that corresponds to the original query retrieves too many documents, then the system uses a search strategy to automatically narrow the query criteria, i.e. make them more specific. In general, a set of alternative result sets will be retrieved.

1) The *subject criteria*, i.e. the terms in the boxes of the *Terms* field are combined using AND and OR in a free-text search of the basic index. If a box contains more than one word, e.g. "Lie group#", then the words are connected with the adjacency operator (W). We call the result **Set 1**.

2) If given, the *Date* criterion is searched for separately, giving **Set 2.1**. *Document type*, *Language* and any other field criteria are then ANDed together in a separate search, giving **Set 2.2**. This is to make it possible later to give a greater weight to *Date*, which tends to be more important than *Document type* and *Language* when a threshold must be achieved. Sets 2.1 and 2.2 are then ANDed together, the result being called **Set 2**, which represents the so-called *formal criteria*.

3) Set 1 is ANDed together with each of the three sets retrieved in step (2), giving **Sets 3.1, 3.2 and 3**.

4 a) If *any of* Sets *3.1, 3.2 or 3* are *above the lower threshold*, then the system performs a new search in which the terms in step (1) are linked by the field co-occurrence operator (L). This retrieves a set of documents (**Set 4a**) in which the specified terms are all in the same field of the basic index, i.e. in Title, Abstract, Controlled Terms or Uncontrolled Terms, and are therefore more likely to be relevant to the user's information need than when terms are simply ANDed across all fields.

b) If Set *4a* is *under the lower threshold*, then Set 3 (the narrowest set above the lower threshold) should be displayed to the user as a *document result index*, as described later, in which it should be explained that an attempt to automatically narrow the query produced a result under the lower threshold. Set 4a is then still available via the Other Results command.

c) If Set *4a* is *above the lower threshold*, then it is ANDed with Set 2.1 (*Date*), giving **Set 4c.1**, and if still above the lower threshold with Set 2 (all formal criteria), giving **Set 4c.2.**

5) a) If *any of* Sets *3.1, 3.2, or 3* are *above the lower threshold*, then the results are displayed in one of three ways:

If steps (3) and (4) have retrieved *more than one set* between the upper and lower thresholds, then the system displays an *other results index*, which is described below.

If steps (3) and (4) have retrieved *exactly one set* within the thresholds, then this set is displayed in a *document result index*, and the other sets, if any, may be explicitly displayed by means of the Other Results command.

Otherwise, all retrieved sets were also *above* the upper threshold, and instead of a search result, the system explains that both the user's original query and all attempts to automatically narrow it did not produce a result less than the specified upper threshold. It also states how many documents were in the smallest set retrieved, and the user is given the choice of inspecting either a specially tailored search hints or the smallest search result.

b) If *all of* Sets *3.1, 3.2, 3* are *below the lower threshold*, then a specially tailored **search hints** window is displayed, which includes possible explanations as to why the search may not have retrieved enough documents, e.g. because one of the sets from steps (1) and (2) is below the lower threshold. If any of the sets in step (2) are empty then an expand list for the corresponding term is automatically displayed. This search hints window also explains the possibilities of misspellings, singular/plural forms, and how these can be handled by truncation and masking. We considered the possibility of performing **automatic truncation** of terms, i.e. of replacing words from the query terms by their stems followed by the symbol that matches any number of characters. Although this would generally improve recall, this would be at the cost of precision. Since precision will very often be of prime importance for mathematicians, we believe it is better to try and teach them how the truncation and masking symbols can be used, so that they can decide how to use them most appropriately for their specific information needs. Finally, the search hints window also explains that the specified terms may be two narrow, and that the New Terms menu can be used to find broader terms or synonyms, e.g. from a frequency analysis, which is automatically displayed if the user has specified a vague information need.

There follows an example of the STN search commands in which this strategy results when applied to the query previously illustrated in this subsection. The example is followed by a Venn diagram which illustrates the relationships between the most important of the result sets that may be retrieved by the above strategy.

```
SEARCH Lie(W)group# AND number(W)theory          Set 1 (L1)
S 1980-1988/PY                                   Set 2.1 (L2)
S english/LA AND journal/DT                      Set 2.2 (L3)
S L2 AND L3                                      Set 2 (L4)
S L1 AND L2   {terms AND period}                 Set 3.1 (L5)
S L1 AND L3   {terms, type AND language}         Set 3.2 (L6)
S L4 AND L3   {terms, period, type & language}   Set 3 (L7)
```

If L5, L6 or L7 is above the lower threshold, then:

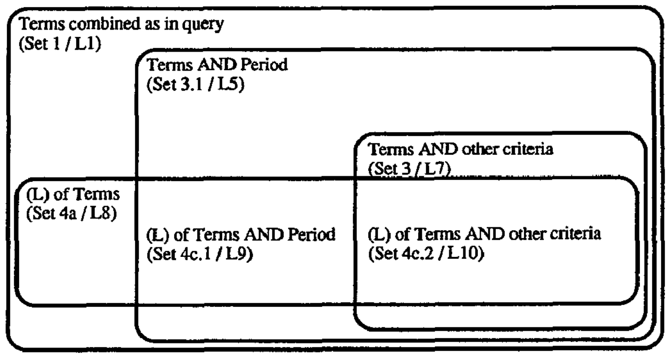
```
SEARCH Lie(W)group#(L)number(W)theory            Set 4a (L8)
```

If L8 is above lower threshold, then:

```
S L8 AND L2   {terms in single fields & period}  Set 4c.1 (L9)
```

If L9 is above lower threshold, then:

```
S L9 AND L3   {fields, period, type & lang.}     Set 4c.2 (L10)
```

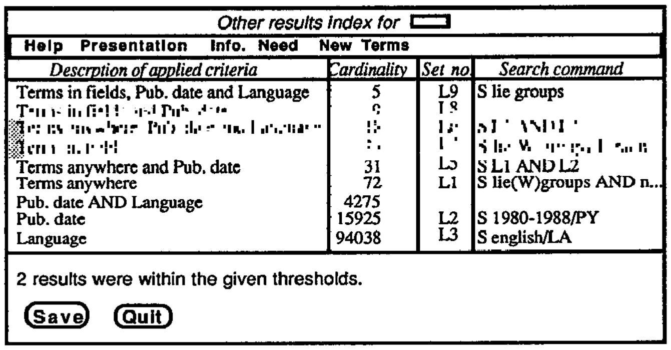


We are aware that many **alternative search strategies** to those described in this paper exist. For example, a domain knowledge base, e.g. a term relationship network, could be used to automatically reformulate queries. Techniques to do this have been described in various expert intermediary systems, e.g. [Larsen 87] and [Croft 86]. The problem in using such techniques in EUROMATH would be acquiring and maintaining the domain knowledge. None of the expert intermediaries we have read of have tackled this problem in a domain similar to that of EUROMATH, and we do not believe that mathematicians are the appropriate user group upon which to test these ideas in practice. To facilitate the possibility of later introducing such alternative techniques as well as the tailoring of the chosen search strategy, it is important that the system is implemented in flexible and easily modified way. The use of **knowledge-based system** techniques can be appropriate in this connection, e.g. the architecture described in [Croft 86].

3.4. Display of Results

The **result of a search** may be displayed in three different ways, as already described under point 5(a) of the search strategy, and as illustrated in the second diagram of subsection 3.2. A **document result index** is displayed as a table, each entry representing one of the documents from a particular result set. A user can select any entry in the table and display the corresponding document.

An **other results index** is also displayed as a table, each entry representing one of the result sets retrieved during the search. The default presentation for novice users displays a description of what each set represents and the set's cardinality. The default presentation for expert users, which is illustrated below, also includes the corresponding search command and the corresponding set number. The entries that represent sets within the specified thresholds are highlighted. The table is ordered in increasing order of cardinality to indicate a form of *ranking*. Any of the sets can be opened, and it is up to the user to decide which sets are of relevance to his information need. Note that the set difference operation is available to create sets without duplicates, as suggested in [Vigil 83], in the case when a user wishes to inspect a set that is a superset of a set he has already inspected.



Other results index for

Help Presentation Info. Need New Terms

Descrption of applied criteria	Cardinality	Set no	Search command
Terms in fields, Pub. date and Language	5	L9	S lie groups
Terms in fields and Pub. date	9	L8	
Terms anywhere, Pub. date and Language	[illegible]	[illegible]	S L[illegible] AND L[illegible]
Terms in fields	[illegible]	[illegible]	S lie(W)groups [illegible]
Terms anywhere and Pub. date	31	L5	S L1 AND L2
Terms anywhere	72	L1	S lie(W)groups AND n...
Pub. date AND Language	4275		
Pub. date	15925	L2	S 1980-1988/PY
Language	94038	L3	S english/LA

2 results were within the given thresholds.

4. Conclusions

Our design has demonstrated that it is feasible to integrate all of the major requirements to the information retrieval facilities of a comprehensive Knowledge Worker Support System in a uniform and comparatively simple user interface. Currently, an implementation of the interpersonal communication facilities of the EUROMATH System is underway within Phase 1 of the EUROMATH Project. Implementation of the remaining facilities, including those described in this paper, is planned to commence in the summer of 1989 and last approximately 2 years. When this implementation is completed, it will be possible to empirically evaluate whether the functionality and user interface that we have proposed provide an adequate solution to supporting our target user group in performing the target tasks.

5. References

[Barbic 85]: "The Type Concept in Office Document Retrieval", F. Barbic & F. Rabitti, Proceedings 1985 International Conference on Very Large Data Bases.

[Bertino 88]: "Query Processing in a Multimedia Document System", E. Bertino et al., ACM Transactions on Office Information Systems, Vol. 6, No. 1, January 1988.

[Croft 86]: "I^{3}R: A New Approach to the Design of Document Retrieval Systems", W.B. Croft & R.H. Thompson, July 1986.

[Draper 88]: "EUROMATH Functional Specification", ed. D. Draper et al., December 1988.

[Furuta 88]: "Interactively editing structured documents", R. Furuta, V. Quint & J. André, Electronic Publishing, Vol. 1, No. 1, April 1988.

[Greenblatt 78]: "A Study of Three Database Query Languages", D. Greenblatt & J. Waxman, in Databases: Improving Usability and Responsiveness, ed. B. Shneiderman, Academic Press, 1978.

[Haines 82]: "Experiences in Training End-User Searches", J.S. Haines, Online, November 1982.

[Harper 85]: "MINSTREL-ODM: A Basic Office Data Model", D.J. Harper et al., Information Processing and Management, Vol. 22, No. 2, 1986.

[Heiler 85]: "G-WHIZ, a Visual Interface for the Functional Model with Recursion", S. Heiler & A. Rosenthal, Proceedings 1985 International Conference on Very Large Data Bases.

[Horak 84]: "An Object-Oriented Office Document Architecture Model of Processing and Interchange of Documents", W. Horak & G. Krönert, ACM-SIGOA Conference on Office Information Systems, Toronto, June 1984.

[Ingwersen 86]: "Cognitive Analysis and the Role of the Intermediary in Information Retrieval", P. Ingwersen, in Intelligent Information Systems, ed. Roy Davies, Ellis Horwood, 1986.

[Ingwersen 88]: "Means to Improve Subject Access and Representation in Modern Information Retrieval", P. Ingwersen & I. Wormell, Libri, Vol. 38, No. 2, 1988.

[Larsen 87]: "Knowledge Representation in IRIS, an Information Retrieval Intermediary System", H.L. Larsen, Proceedings 7th International Workshop on Expert Systems and their Applications, Avignon, May 1987.

[McAlpine 85]: "A Graphical User Interface to an Office Filing Facility", G. McAlpine & P. Hougaard, ESPRIT Technical Week, North Holland, 1985.

[McAlpine 88]: "Principles and Techniques for the Graphical Presentation of Information for Browsing of Information Bases", G. McAlpine, NordData 88.

[Morrissey 82]: "An Intelligent Terminal for Implementing Relevance Feedback on Large Operational Retrieval Systems", J. Morrissey, Conference on Research and Development in Information Retrieval, Berlin, 1982.

[Schmid 75]: "On the Semantics of the Relational Data Model", H.A. Schmid & J.R. Swenson, Proceedings 1975 ACM SIGMOD International Conference on Management of Data.

[Smith 82]: "Designing the Star User Interface", D.C. Smith et al., Byte, Vol. 7, No. 4, 1982.

[van Rijsbergen 86] "A Non-classical Logic for Information Retrieval", Computer Journal, Vol. 29, No. 6, December 1986.

[Vigil 83]: "Analytical methods for online searching", P.J. Vigil, Online Review, Vol. 7, No. 6, 1983.

[Wegner 87]: "The Object-Oriented Classification Paradigm", P. Wegner, in Research Directions in Object-Oriented Programming, B. Shriver & P. Wegner (ed.), MIT Press, 1987.

[Zloof 81]: "QBE/OBE: A Language for Office and Business Automation", M. Zloof, Computer, Vol. 14, No. 5, May 1981.