Designing a Macintosh Interface to a Mainframe Database

William H. Benson and John L. McCarthy
Computer Science Research Department
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

Abstract

Mainframe database systems are typically accessed by terminal emulation, effectively ignoring much of what makes workstations attractive. MacMIST provides an interface based on standard Macintosh paradigms, as well as application-specific ones, to access a material properties database running on a remote IBM 3084/3090. This work is part of a large project called MIST (Materials Information for Science and Technology) to develop a prototype information system for materials science. In the future, this architecture is expected to provide the basis for a common interface to a network of quasi-autonomous databases.

MacMIST illustrates why context dependent information is needed as part of the interface. MIST provides not only raw data, but also extensive metadata — including a thesaurus and glossary, knowledge of class hierarchies, and summaries of what data is currently available. MacMIST supplements users' a priori knowledge of context and meaning by incorporating such metadata into the interface dynamically. Use of caching and embedded menus facilitates fast and efficient access.

1.0. Introduction

Personal workstations are becoming widespread and increasingly popular. Users quickly become accustomed to the inherent responsiveness and availability of a local processor, and come to expect ever greater capabilities from new software. Experience with personal computing tends to lead to heightened expectations for any software new to the user, even when, as in the present case, that software is actually a large database system running on a remote mainframe. Such systems are typically accessed by terminal emulation, effectively ignoring much of what makes workstations attractive, and limiting the user interface to the relatively primitive and awkward interactions standard for mainframe-terminal communications.

Since the user interface is typically conceived as a separate component of a database system, it is natural to relegate the mainframe to the role of query server and locate the interface entirely on a workstation (Fig. 1). We discuss some of the issues and opportunities presented by this approach in the context of a specific implementation on the Macintosh of an interface to a specific database system.

The architecture follows a client-server model. Query formulation takes place on the Macintosh. Queries are sent to the mainframe to be executed, and search results are downloaded back to the Macintosh. All transactions are cached locally to enhance performance.

This work is part of a large project called MIST (Materials Information for Science and Technology) to develop an information system for materials engineers and researchers. This paper describes the project from the point of view of the user interface; for a more comprehensive description, see [1,2,3].

2.0. MIST Project Synopsis

MIST is a prototype material properties data system. It has been developed by Lawrence Berkeley Laboratory for the U.S. Department of Energy, with support from the National Bureau of Standards and Sandia National Laboratories.

The immediate short range goal is to increase scientific research and development productivity: by providing tools for expediting dissemination of new results, improving data quality, promoting data sharing, reducing duplication of effort, and identifying significant gaps in current knowledge.

A longer range goal is to make MIST a distributed system. The approach that seems most promising at this time is to provide users access to a network of quasi-autonomous databases, each specializing in a limited subject area, but unified by a common data directory, consistent nomenclature, and compatible database structures [3].

![Client-Server Architecture](image)
3.0. Users, User Needs, and User Access

Users are engineers and researchers in Materials Science who want to search and retrieve numeric data about materials and properties.

For example, there are two generic types of queries for material properties databases. On one hand, users may want to retrieve all the property data available for a given material, much as they currently do using printed handbooks. On the other hand, they may want to know all the materials satisfying certain property constraints, such as "all metal alloys with melting points greater than 1300 degrees Centigrade and densities less than 7.0 g/cc".

Traditional mainframe interfaces to comparable information retrieval systems have been so hard to use that people rely on intermediaries -- information retrieval specialists -- to perform searches for them. A familiar example is the on-line version of the Official Airline Guide. Specialists, such as Travel Agents or department secretaries, assist travellers with queries, enter search commands, and interpret flight listings.

A workstation platform presents new opportunities to make the interface more intuitive and more intelligent.

A sufficiently intuitive interface can enable users to access information systems directly, without intermediaries, and attract those users who would otherwise not use such a system at all. Moreover, it is expected that many users will access the MIST database infrequently or on an irregular basis. By making the interface more intuitive, users will be encouraged to participate if they see that little or no investment in learning or re-learning is required.

A more intelligent interface can also help users formulate queries and interpret search results by making the extensive thesaurus component of MIST (described in section 4.0) more accessible.

4.0. Database Components

The principal contents of MIST are scientific data and metadata -- data about the data, such as descriptions, measurement units, etc. A typical example is test results -- measurements and observations describing the behavior and characteristics of physical materials as various parameters are varied. The data are currently drawn from tables and graphs in printed handbooks and machine-readable sources. In the future, user contributed data will also be included.

The emphasis on archival data implies that the main focus of the interface should be on queries and data retrieval. The interface supports data display, but currently does not provide for updating or data entry. The latter functions are handled separately, as they have requirements distinct from retrieval.

Tables and graphs from scientific and engineering handbooks are highly structured and the text and numeric information they contain is densely packed. This compact layout underscores the point that the data are not simply numbers in isolation, but rather a structured organization of observed values and descriptors. This metadata, such as source, footnotes, units, and independent variables must be carried along with the data in order for the numbers to be meaningful.

In MIST, tables and graphs are decomposed into datasets, consisting of both data and metadata bundled together, and are stored in the database in canonical form. The datasets contain both descriptive metadata -- such as table or figure title, source, footnotes, references, and other auxiliary information -- and vectors of property and variable values for some given material(s). Datasets commonly contain some twenty or thirty such covariates (properties, independent and dependent variables). The data may be drawn from many sources, each using different nomenclature and covering different subsets of materials and properties. Datasets are stored in canonical form so that, in principle, data is comparable across different sources as well as from one form of representation (e.g. tables versus graphs) to another. A workstation based interface presents a number of opportunities to facilitate making such comparisons.

In addition to datasets, whole tables and graphs are stored in separate data structures in such a way that a facsimile of the original can be reconstructed. There is a practical advantage -- the canonical dataset representation can be derived by an automatic algorithm from the structured representation of a table or graph (but not vice versa). In addition there are important advantages for the user -- the security of familiar printed formats from the handbooks, and the authority and credibility of such documentation.

The database also includes an extensive thesaurus containing definitions and descriptions for material names, properties, variables, units, and other terms. Thesaurus entries are cross referenced to broader terms, narrower terms, related terms, etc. The thesaurus supports automatic indexing of database-specific nomenclature to standardize terms, as well as access to definitions and cross references.

Users may need to look up information at any time, but especially when they need to know which terms to use (from a controlled word vocabulary of material and property names) to compose a query. This basic need is addressed by organizing the large number of material and property names into more manageable class hierarchies, as reflected by the broader terms and narrower terms relations in the thesaurus.

5.0 Prototype Database Implementation

The prototype MIST system has been implemented on SPIRES, a mature fourth generation database management system (DBMS) developed at Stanford University and presently running at some thirty other sites. SPIRES currently runs on IBM (3084/3090) mainframes under several different operating systems; a C version currently under development is intended to make SPIRES available on a much wider variety of systems. It is envisioned that MIST will ultimately encompass distributed heterogeneous DBMSs running at remote hosts.

In addition to a workstation front end, SPIRES itself supports three other general approaches to user interface design: native query language, full screen menus, and command files.

Query language. The SPIRES query language can be illustrated by an example, such as

\texttt{find (material=aluminum) and (property=density)}
which searches for all datasets with both aluminum and density data. Other Boolean and relational operators can also be used. The difficulties with the Boolean request form for users typing at a terminal are well known [4]. However, it is well suited for workstation-mainframe communications following a client-server model. Similar message-passing architectures are already used by several commercial DBMSs with distributed capabilities (e.g. ORACLE, Ingres). In these systems, front end applications issue requests to database servers in the form of SQL statements.

Full screen menus. SPIRES provides a mainframe implementation of full screen menus for cursor-addressable terminals, such as the DEC VT100. A full screen menu interface was considered, but was unnecessarily restrictive and not sufficiently responsive over dialup communication lines.

Command files. Many interface designs can be implemented with command files. In particular, a line oriented menu system has been implemented on SPIRES in the mainframe environment in a parallel part of the MIST development effort [5].

Some insight into user interface needs for generic data retrieval can be gained by looking at what SPIRES provides to supplement its raw query language. There is the notion of a "search result", and a "stack". Data retrieval is initiated by issuing a find command, as above, which returns a list of record pointers as the search result. The search result can be expanded or narrowed according to additional search criteria. The stack is another list of record pointers which can also be sorted. Selected records can be copied from the search result to the stack. Records from either list can be displayed at the terminal. There are also ways to save and restore named sets of query commands (macros), search results, and stacks so that earlier work does not need to be repeated.

Search result and stack lists have two advantages for the user. They provide a workspace area in which records can be examined, discarded, re-arranged, etc.; and they enhance the responsiveness of display requests since the physical locations of the data records are already known.

In sections 7.3 and 7.4, we discuss how these insights can be applied to MacMIST.

6.0. Why Use Macintosh?

The major reasons for choosing a Macintosh platform are

- to capitalize on users' familiarity with standard user interface conventions. One reason for the success of the Macintosh is that users can get started with new software immediately, often without printed documentation, since most software does follow standard conventions.

- to take advantage of the substantial thought that has gone into developing the Macintosh user interface guidelines, design principles, and philosophy [6]. Some of the most influential aspects are: direct manipulation; modelessness; the basic paradigm for manipulating information - that selecting information precedes choosing an action; and that users can rely on recognition instead of recall.

- the availability of toolbox support for the standard user interface, and software development environments that facilitate rapid prototyping. While the Macintosh may have deserved the reputation at one time as difficult for developers, there is now a wide choice of compilers and development tools. Another indication of a favorable environment is that an increasing number of commercial products are first developed on the Macintosh and later ported to other workstations. A similar rationale applies to the present project. A quick, even though incomplete, implementation is needed to establish proof of principle and guide further design.

7.0. Design Considerations and Strategies for MacMIST

This section outlines a conceptual model for user interaction, which is then described in more detail in section 8. Other topics discussed are retaining context, lists as workspaces, and routine caching of all mainframe transactions.

7.1 Conceptual Model. MIST users want to search for relevant information, compare and analyze search results, and document findings. A systematic approach toward satisfying these needs is to proceed step by step through a set of specific query and retrieval activities. Fig. 2 presents a number of such activities organized into phases or stages.

| Identify material and property names from controlled word vocabulary |
| Formulate query (select names, values) |
| Issue query |
| Retrieve summary info, number of records found |
| Reformulate query if too few or too many records found |
| Trade-off level of detail vs. download time |
| Browse summary info (property names only) |
| Retrieve selected datasets |
| Browse formatted display (full datasets) |
| Cut & paste to analysis programs |

Figure 2. Query Life Cycle Phases

The diagram is intended to suggest an analogy with Software Engineering life cycle concepts. It points out that distinct phases can be identified -- with specific activities in each phase -- and that the phases can be arranged in a sequence. With queries, as in software development, it is important to be able to move from one phase to another, either forward or backward, and review progress at intermediate steps.

There also is a notion of reuse analogous to software reuse. Query components or extracts from search results can be used as templates for further query modification. Moreover, the successive refinement of a series of queries, each based on feedback from previous ones, follows much the same paradigm as the software engineering concept of rapid prototyping.

The software life cycle has been described as the most important unifying concept for integrating software development techniques [7]. Continuing the analogy, we have designed the interface to promote the query life cycle
as a conceptual model for user interaction. Users satisfy a major information retrieval goal by achieving well defined sub goals in successive stages, with some significant portion of the work accomplished in each stage.

The interface reinforces this notion of a sequence of phases by presenting the user with a regular configuration of stacked windows, each slightly offset from and overlapping the ones behind (Fig. 3). You proceed through the windows as you proceed through the phases of query formulation and retrieval. A window made active is brought to the front, but the offset visually maintains the window’s relative position in the stack. A menu command is provided to quickly return to the “home” configuration.

Tombaugh et al. [8] investigated such a window design, pointing out that overlapping windows can provide both perceptual and procedural stacking. They were concerned with reading lengthy text; however their findings indicate that such multi-window designs can significantly help users find information they had in front of them just a few moments ago to relocate themselves in a previous context (e.g. in a different stage).

The help menu (section 9.2) also tries to guide the user through a sequence of activities by presenting them as a list of numbered items – focusing attention only on those that are currently possible. Higher numbered steps, which depend on lower numbered ones, are dimmed if the earlier steps have not yet been taken (Fig. 8).

Phase related activities are described further in section 8.

7.2. Retaining Context. One of the major advantages of a multi-window design is that the windows can retain context.

We have emphasized the idea of passing through a sequence of phases – because this helps to establish the conceptual model; but this does not imply that all selections must be performed in a rigid order. You can select any window to revisit a previous context and make selections anywhere in any order.

In particular, this allows incremental query modification. This capability is not typically provided by DBMSs, even on the Macintosh, but it is especially helpful in coping with one of the major problems with Boolean search strategies: null output and output overload [4].

Another advantage of keeping context is that users can interrupt any activity, pause to look up terms or other metadata in the thesaurus, and return without losing work in progress or losing their place.

7.3. Lists as Workspaces. Observations about DBMSs such as SPIRES (section 5) indicate that the notion of a workspace where search results can be manipulated – in particular, the workspace as a list of items – is easy to use and understand.

A workstation platform allows these notions to be extended throughout the interface and made more coherent and consistent: first, by providing separate workspaces for separate activities; second, so that they are made visible and tangible. A workspace occupies real space within some window on the screen. This reduces cognitive effort, since lists can actually be seen instead of imagined, and makes list items accessible for direct manipulation. In particular, users can specify list items for query or retrieval by selecting them directly.

The notion of a list can be generalized or extended to meet specific needs in each stage. In stage 1, an outline format for material and property classes helps users navigate through classification hierarchies (Fig. 3). In stage 2, sub lists can show the set of allowable property values or numeric ranges for a given property. In stages 3 and 4, two dimensional lists provide a spreadsheet-like format for comparing materials and covariates together (Fig. 4, 5). In stages 5 and 6, a formatted report can be considered as a scrollable list of text lines (Fig. 6).

7.4. Caching. A second observation from DBMSs such as SPIRES (section 5) is that it pays to keep some kind of history of both user and system activities to avoid the time and effort of repeating previous work.

Caching is a uniform approach to providing these benefits, and it can be used in many different situations. Performance gains can be dramatic when the cache is located at the workstation itself, in memory or on disk. By routinely caching mainframe transactions in local storage, subsequent requests for the same information can be satisfied immediately, completely avoiding mainframe access and transmission delays.

MacMIST caching is automatic and transparent, unlike the SPIRES mainframe environment where a user must explicitly save, name, and recall cache, search results, or stacks. Moreover, by retaining the cache on a local disk from one session to the next, the user can quickly restore the context from a previous session, resuming from the point where work was suspended.

Even with a cache, the data ultimately must come from the mainframe. In this case, the user can be given control to decide what trade-off to make between transmission time and which data and how much to retrieve. A sense of being in control may make slow mainframe retrievals more acceptable.

8.0. Phase Related Activities

The activities in each of the six stages of query formulation and data retrieval, as outlined in Fig. 2., are described in more detail below.

8.1. Stage 1: Identify Material and Property Names. A hierarchical classification scheme is used to organize the potentially large number of material and property names. Users navigate these classification hierarchies to identify material and property names of interest.

Two list of material classes and property classes are shown as partially expanded outlines in Fig. 3. Materials and properties may be listed under multiple classes and sub classes (i.e., a term may have more than one broader term). Classes can be expanded or contracted through popup menus which appear when the mouse is pressed over a highlighted list item. Classes are expanded into subclasses, perhaps several times, until a set of material or property names is reached. At this point, the names are entered in a separate workspace list directly below the corresponding class list as shown in the lower half of Fig. 3.
Two special classes are assigned dynamically outside the regular hierarchy – all materials with data for a given property, and all properties for a given material. These are accessed, one at a time, from the top of each list. Given a material name, for example, a user wants to know what property data is available to avoid a null search result. This is important because data is very sparse in the total matrix of materials and properties. See Fig. 3, where selecting "aisi 1025" has redefined the property class at the top of the list to be "all 'aisi 1025' properties".

8.2. Stage 2: Formulate the Query. The main focus of the interface is query formulation and data retrieval. Queries consist of Boolean combinations of materials and properties; properties may be further qualified by values or ranges.

Materials, such as "aisi 1025" and properties, such as "thexp" in Fig. 3, are selected to participate in a query expression by clicking on them with the mouse. A query is constructed automatically in two steps. First, a subexpression is formed by combining all materials with OR. In addition, each property forms its own subexpression. Second, all subexpressions are combined with AND. For example, (material=aisi 1025 OR material=a36) AND (property=thexp) AND (property=specific heat). The rationale for this method of query construction is discussed in section 10.

Selections can be modified at any time, since the query command is not constructed until the user explicitly requests retrieval, and since the selected items are never automatically de-selected.

8.3. Stage 3: Issue Query. The Retrieve menu is used to issue the query, which, if not found in the cache, is sent to the mainframe to be executed. Issuing the query brings the next window (Dataset Summary) to the front. The search results consist of summary information, which can be downloaded in one of several formats. The user can trade-off transmission time against amount of detail. The formats respect the canonical representation, keep both data and metadata together, and facilitate side-by-side comparisons of covariates.

8.4. Stage 4: Browse Summary Information. The number of datasets satisfying the query is always presented first so that the user has an opportunity to cancel the transaction. The briefest summary shows only the title of the table or graph from which the datasets were derived. A slightly more detailed summary is given by including the list of covariates for which data is available. An example is shown in Fig. 4, where each row corresponds to a dataset. Some covariates that can be recognized are "temperature", "form", and "thermal coefficient of expansion". Dataset titles (truncated) are in the first column. Other covariates and datasets can be seen by scrolling vertically (for datasets) or horizontally (for covariates). The narrower scroll bar at the top adjusts the column width.

A more detailed summary can include values or ranges as well as the names of all covariates. An example is shown in Fig. 5, where, in order to facilitate side-by-side comparisons, each column corresponds to a dataset. The window contents can be scrolled, and rows and columns can be sorted and rearranged to make comparisons easier.

From these summaries (titles, property names, etc.) users can decide whether to abandon the current query and try another one, or proceed to download complete information about some or all of the datasets found.
### Dataset Summary

<table>
<thead>
<tr>
<th>Dataset Titles</th>
<th>Covariates</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIGURE 2.2.1.0. Effect of temperature</td>
<td>thermal coefficient of expansion</td>
</tr>
<tr>
<td>FIGURE 2.2.1.0. Effect of temperature</td>
<td>thermal coefficient of expansion</td>
</tr>
<tr>
<td>FIGURE 2.2.1.0. Effect of temperature</td>
<td>thermal coefficient of expansion</td>
</tr>
<tr>
<td>FIGURE 2.2.1.0. Effect of basis</td>
<td>thermal coefficient of expansion</td>
</tr>
</tbody>
</table>

| TABLE 2.2.1.0(a). Material specification | form |

Figure 4. Dataset Summary Window

### Dataset Legend

<table>
<thead>
<tr>
<th>EXHIBIT TITLE:</th>
<th>FIG. 3.0322 EFFECTS OF LOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE:</td>
<td>Chapter 2-1 501 (revised 3/78), Chapter 2-1 303 (revised 3/78), ASME, ASME</td>
</tr>
<tr>
<td>MATERIAL (UNS):</td>
<td>17-4PH (S17400), 304 (S30400)</td>
</tr>
<tr>
<td>DATABASE (ID):</td>
<td>ASME, ASME</td>
</tr>
<tr>
<td>FORM (text):</td>
<td>bar, sheet, strip, plate</td>
</tr>
<tr>
<td>CONDITION (text):</td>
<td>annealed</td>
</tr>
<tr>
<td>HEAT TREATMENT (text):</td>
<td>1575 F, 4 hr, OQ + 4hr, 450 F, 1515 F, OQ + Temper</td>
</tr>
<tr>
<td>COMPOSITION:</td>
<td>Fe-0.4C-1/8Ni-0.8Cr-0.25Mo -</td>
</tr>
<tr>
<td>DIAM (in.):</td>
<td>2.50</td>
</tr>
<tr>
<td>TEMP (degF):</td>
<td>77.32 - 102.4</td>
</tr>
<tr>
<td>FCY (ksi):</td>
<td>80.2</td>
</tr>
<tr>
<td>CYS (ksi):</td>
<td>98.1</td>
</tr>
<tr>
<td>DENSITY (lb/in^3):</td>
<td>0.280</td>
</tr>
<tr>
<td>HARDNESS (Re):</td>
<td>51.00944 - 64.08164</td>
</tr>
</tbody>
</table>

Figure 5. Comparing Datasets Side-By-Side

### Full Dataset Display

| MATERIAL (UNS): | R1025 (G10250) |
| DATABASE (ID): | AMS (AMS 2112.2.1.0(b) 15/01/83) R00182, 05/07/87 awk |
| SOURCE: | Chapter 2 (revised 6/01/89), Table 2.2.1.0(b) |
| EXHIBIT TITLE: | TABLE 2.2.1.0(b). Design Mechanical and Physical Properties of R1025 Carbon Steel |
| BASIS (text): | S |
| THICK (text): | 5 |

<table>
<thead>
<tr>
<th>Line</th>
<th>Num</th>
<th>Cond text</th>
<th>Form text</th>
<th>FTUL ksi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cold rolled</td>
<td>Sheet, Strip</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Normalized</td>
<td>Tubing</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>All</td>
<td>Bars</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>Num</td>
<td>F2U, LT ksi</td>
<td>F2U, ST ksi</td>
<td>F2U, LR ksi</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>1</td>
<td>55</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>36</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>55</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>FBU, 1.5 ksi</th>
<th>FBU, 2.0 ksi</th>
<th>F2B, 1.5 ksi</th>
<th>F2B, 2.0 ksi</th>
<th>EL, L ksi</th>
<th>EL, LT ksi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num</td>
<td>FBU, 1.5 ksi</td>
<td>FBU, 2.0 ksi</td>
<td>F2B, 1.5 ksi</td>
<td>F2B, 2.0 ksi</td>
<td>EL, L ksi</td>
<td>EL, LT ksi</td>
</tr>
</tbody>
</table>

Figure 6. Standard Dataset Report Format
AISI 1025 is an excellent general purpose steel for the majority of shop requirements, including jigs, fixtures, prototype mockups, low torque shafting, and other applications. It is not generally classed as an airframe structural steel. However, it is available in aircraft quality as well as commercial quality.

**Figure 7. Thesaurus Information**

**Figure 8. A Conversation Model for Help**

8.5. Stages 5 and 6: Full Dataset Retrieval. The dataset titles to be retrieved are selected with the mouse, and the Retrieve menu is used again to download pre-formatted reports of the full datasets or the original exhibit facsimiles. Output can be viewed in a text window with scroll back control. A menu command can be used to print one or more datasets. Standard cut and paste techniques can be used to transfer some or all of the data to other programs for further analysis.

For example, the first and fourth datasets in Fig. 4., as indicated by the highlighting, have been selected for retrieval. A portion of a full dataset report is shown in Fig. 6.

8.9. Phase Independent Activities

As in software development, there are certain activities users may want to perform throughout the life cycle. In particular, users may want to consult the thesaurus about subject matter, or access a "help" function regarding the interface itself.

9.1. Thesaurus Consultation. A list of user selections (material and property names, class names, abbreviations, etc.) is accumulated in a workspace in a separate (Terms) window. Users can consult the thesaurus at any time by selecting one of the terms from this window. A popup menu with retrieval commands appears when the mouse is pressed over the highlighted term.

Information about the term is displayed as a partially expanded outline (Broader Terms, Narrower Terms, Description, etc.) similar to the format for material and property class hierarchies in stage 1. Fig. 7 shows the thesaurus entry for AISI 1025 under its standard name of g10250. The Broader Terms class has been expanded to show carbon steels. An expanded term can be looked up in the same way—which provides a kind of hypertext browsing capability.

Since all the windows retain context, users can always interrupt an activity to look up a term, and resume without losing their place.

9.2. Help. There are reports in the literature that although help functions are widely provided, they are largely unused. Ridgway [9] analyzes the situation as follows. Help functions usually follow a book model, where the user looks up topics in an index. Some of the problematic characteristics of the book model are

- help is usually obtained by request;
- searching an index disrupts the flow of thought;
- new help screen hierarchies can be disorienting;
- users quickly get impatient when help is not helpful.

It is conjectured that what users really want is a conversational model, and in particular one that understands the current context—since they are already accustomed to a conversational metaphor for the user.
interface. This suggests that a program should volunteer help without waiting to be asked. Ridgway points out that error alerts and warnings are often reported this way and are well received by users.

Another approach along these lines is to put nested questions and answers into hierarchical menus. Users can pick their question from the menu, displaying a checklist of tasks to do, which in turn may lead to picking further questions from the submenus (Fig. 8).

This is non-intrusive, but still suggests a conversation, and knows something about the current context. Menu items can be enabled or disabled according to whether an action can be performed yet (e.g. a query cannot be issued until material or property names are selected); or the menu items themselves can be replaced with different text depending on the context. The menu items are only messages — not commands — although they could give more detailed information in a separate window.

Although putting messages in menus is non-standard usage, it is fairly common in game software.

### 10.0. Augmenting Boolean Search Strategies

Despite numerous alternatives in the literature, nearly all information systems and DBMSs (including SPIRES) continue to use pure Boolean search strategies. Problems with Boolean strategies and some solutions are discussed by Cooper [4]. This discussion is briefly summarized below.

Some of the well known problems are that Boolean expressions can be tricky to construct; that it is difficult to formulate queries precisely enough to avoid null output or output overload; and that Boolean logic assumes all search criteria are equally important.

In teaching students how to use Boolean retrieval systems, it is often suggested that they develop a multi-faceted classification of their information needs. This involves building a list of search terms for each facet; and forming a Boolean search expression by systematically ORing within the lists and ANDing between them. This by no means generates all possible queries, but in practice satisfies most information needs. Furthermore, given the facets and selected terms, it is simple to construct this form of Boolean query automatically.

MacMIST uses this approach to construct Boolean queries. For MIST, all materials can be considered to constitute one facet, and each property constitutes an additional facet.

This approach could be augmented, if needed, by providing set operations (union, intersection, difference) on the search results from different queries. Set operations are provided, for example, in the FIRSTUSER interface [10].

There are many alternatives proposed for the other two problems. Some of these could be implemented within a workstation based interface even though the mainframe database only understands pure Boolean expressions.

As one example, queries could be developed such that the search results would contain any record matching at least one of the facets. A workstation based interface could then rank the records according to how many facets were matched. Users could scan from the top to look first at those matching the most facets, and stop when the records were no longer relevant.

### 11.0. Conclusion

MacMIST illustrates a number of points about the user interfaces to information retrieval systems in general, and front end interfaces to mainframe database systems in particular:

- **why context dependent information is needed as part of the interface.** The information needed to compose queries, compare search results, and access the thesaurus ultimately comes from the mainframe database. This includes lists of search terms for which data is currently available, which property is available for which materials and vice versa, and a context of broader, narrower, and related terms to enable hypertext browsing in the thesaurus. Incorporating this metadata into the interface dynamically makes it available and accessible at the time it is needed.

- **how to get efficiency in such situations.** Efficiency is achieved: through a direct manipulation style of interaction, which implies that any output that appears on the screen can be used as input for some other operation; by embedded menus; and through extensive caching.

- **how retaining context can reduce cognitive effort.** By retaining context, users can make selections in any order, modify queries incrementally, and browse the thesaurus without losing work in progress or losing their place.

- **why search terms and search results should be shown as visible, tangible objects.** Search terms and search results are the focus of the users' attention. Presenting them as concrete objects makes them accessible to direct manipulation, and allow users to rely on recognition instead of recall.

- **how a conceptual model of interaction can be conveyed to the user.** Users will always develop some kind of conceptual model, but it may be different from what the designer had in mind. The two can be better aligned by providing a specific location (window, workspace, list) for each distinct activity; providing visual cues for any natural sequence of activities (overlapping windows for perceptual and procedural stacking); and reinforcing similarities in objects and operations by using similar structures (lists, outlines, tables, lines of text) and user action protocols (activating embedded menus over selected items (Fig. 8)).

- **whether a conversation model of help is useful or feasible.** We have raised the issue, and provided an implementation that users can test and compare with conventional models.
Acknowledgment

This work was supported in part by Sandia National Laboratory through the U.S. Department of Energy under Contract No. DE-AC04-76DP00789, and by the Applied Mathematical Sciences Research Program of Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098. Sandia provided the impetus for developing a Macintosh interface. Their encouragement and support is gratefully acknowledged.

References


