P-9

Reprinted from
Journal of the American Society for Information Science
July-August 1973 Vol. 24, No. 4

Information Organization for Interactive Use: Design Implications in Data-Base Systems

The main purpose of this paper is to seek classes of design implications in data-base oriented languages. Its scope is not limited to library automation, but extends to the problem of scientific and business databases. As a point of practical reference it reviews the lessons drawn from the application of the DIRAC-1 language to a variety of problems at Stanford. The second version of DIRAC, which is currently used in the organization and utilization of a geoscience data-base, is described. An attempt is made to relate the observation of these real-life systems to general characteristics and leads to hypotheses concerning future design trends. In this context two points are regarded as especially important, namely the handling of structure specification and the need to view the user as an operator rather than as a passive recipient of information.

JACQUES F. VALLEE

Institute for the Future
Menlo Park, California 94205

GERALD L. ASKEVOLD

Dept. of Applied Earth Sciences,
Stanford University

• Outline

This presentation discusses information systems in the context of interactive use. It consists of two related lines of comments, each dealing with a certain type of implication, as viewed by the designer in one case, by the user in the other case.

In the context of this paper we regard as an "information system" any organization that provides data upon request from human users. We deal more specifically with organizations that use the computer as the medium of implementation and, among these, our attention centers on data-base management services that allow their users to create, update and query their own data files. Such characteristics were met in the application at Stanford University of two successive versions of a generalized language named DIRAC.

In such a system the user is typically a non-programmer with a high-level problem in mind who is investing considerable time in a series of transactions with the system. (This usage contrasts with that of the researcher who makes use of a program in the batch or remote-batch environment.) Typical of the problems met by such users are applications of the DIRAC-2 language in the creation and management of mineral resource data-bases. The second part of this presentation draws from this experience, which includes the ability to dynamically retain, name and perform logical operations upon subsets of a main file while searching.

In the course of problem-solving using an interactive device the language interface may create motivation or, on the contrary, generate frustration. We will show how this relationship can force both the system and its user to evolve, a process that was clear in the early stages of development of DIRAC: This will be illustrated by observation of a prototype network serving the American astronomy community. In this context it is of interest to ask what the implications of the *use* of the system were on the design changes that took place: What were the constraints that made feedback desirable? What were the conditions that made it possible? And what guidelines emerged for future design phases?

Critical observation of the evolution of DIRAC and similar systems leads to an essential question: Now that we can build generalized systems of this type, and now that we realize their major design implications, to what extent can we say that they answer the information needs of scientists? Have we identified all the properties of the interaction? In order to answer this question we must rephrase it at several levels. It will be useful, for instance, to re-examine the somewhat naive concept of information as a quantity stored in the computer and simply retrievable at will by a passive user. The last part of this presentation will offer practical examples where this naive notion proves inadequate.

If a great part of the intrinsic structure of the data is undiscovered, this implies that the retrieval language must supply facilities for the manipulation of information at many levels in more ways than are currently available. It is felt that some of the problems that are apparent from existing experience with data-base systems will only find solutions in terms of systems that allow dynamic structuring capabilities. Some ideas are proposed to illustrate this process. Throughout this discussion, the word 'system' will apply to an organization of resources designed to support a given process, and the word 'language' will designate the set of words that are used in the communication between this system and a human operator, together with the syntactic and semantic rules that describe the process.

• Definitions

This discussion will be restricted to those information systems that exhibit the following features:

1. Decisions are made on the basis of the data provided. (Examples: retrieval of a particular book, modification of a program, etc.);
2. The ratio of text-editing to programming is over fifty percent. In other words, it is equally used for on-line document preparation and for programming;
3. Under typical operating conditions, users have no contact with the computer except via remote terminals;
4. User data are available on-line without human intervention;
5. Problems are described to the system in a user-oriented language;
6. Communities can share data and services through the system;
7. Development trend is towards multi-dimensional displays and a command language capable of handling graph-structured text.

There are currently fifty to one hundred such systems in Research and Industry. It is of special interest to restrict our attention to a particular sub-category that has not received a great deal of attention outside the literature of library automation, namely those data-base systems that have interactive capabilities. (In addition to the above characteristics, interactive data-base systems must provide for the management at the physical level and the retrieval at the logical level of structured data files.)

This presentation will attempt to: 1) Describe the design and operation of such a system in general terms; 2) Identify its components; 3) Analyze its design implications; 4) Examine its underlying principles; and 5) Discuss briefly future developments and their implications.

Users of such interactive systems are allowed to create, update and interrogate their own information files, and this interrogation may take the form of simple (linear) search through records and catalogues, manipulation of research files with evolving structure, or even complex operations on distributed data-bases.

Typical of the new generation of data-base systems is the fact that no programmer intervention is required in any of the above operations. In other words, we shall only be concerned with those systems that have been designed in such a way that programming concepts would become entirely transparent to the on-line user.

There are approximately half-a-dozen such systems on the U.S. market, many more in private use, although it is difficult to compare their capabilities from the point of view of interactive utilization: they make use of widely varying concepts of file structure and command language design.

• Problem of Interfaces

It is frequently useful to phrase information processing problems in terms of interfaces. Given the user world on one hand and the set of machine representations on the other, language interfaces can be identified at a variety of levels: symbolic languages and assemblers, algorithmic languages and their compilers have been defined and their implications discussed.

The place of information-oriented languages in this hierarchy of systems is not always clear. Some text-editors, for instance, contain compilers as a subset, and users may have to be aware of programming concepts when using them. Some advantages of this approach in improving the performance of software developers have been reported (*6*). Designers of data-base systems usually attempt to avoid any mixing of multi-level concepts, but some languages do include small compilers within the scheme of their interpretive structure. (DIRAC solved this problem through a FORTRAN interface allowing users to provide their own code when they wished to overlay the built-in computational facilities made available as a default.)

Such systems have implications that are much farther-reaching than previous types of software. These implications are important to the designer, to computer professionals in general, to users, and, ultimately, to the way people think about their information problems.

• Anatomy of an Information System

Much of the discussion that follows is based on the experience with two languages, DIRAC-1 and DIRAC-2, whose applications at Stanford University since 1969 have covered the fields of astronomy (*7*), hematology (*5*), blood bank design, surgery file organization, radiotherapy (*4*), administration, etc. The results of these applications have been summarized elsewhere (*3*). The opportunity to carry out a long-term analysis of these implications at Stanford was lost when the DIRAC-1 group (and much of its early language design concept) was absorbed by another project with quite different objectives. A second version of the language, DIRAC-2, was later developed privately. One specialized version of this system has been operational on the IBM 360/67 computer at Stanford since early 1971. It is currently used in the organization of a mineral resources data-base that will be described in special detail.

The following is an illustration of the use of DIRAC-1 in an astronomical search. The user in this case is a scientist interrogating a catalog of supernovae. His first question restricts the search to 23 stars in the Virgo cluster that have been definitely recognized as supernovae. Among these, he asks, how many have a known recession velocity between one and two thousand km/sec? The answer is eleven. In this subset, is there one that has been referenced in the literature under a publication from Mount Wilson Observatory? The system locates one star, supernova number 1901b, and the literature concerning this star is immediately printed, together with some selected parameters.

In this illustration from an actual printout, the prompt from DIRAC is given in full as the word "ACTION" followed by a carriage return and colon. What appears on that line is the retrieval command as typed by the astronomer. Capitalized words are DIRAC keywords. The answer is given after printing of a separation line for readability.

```
------------
ACTION
:    Cluster CONTAINS Virgo AND SN DOES NOT CONTAIN s
     END
------------
   23 RECORDS SELECTED
------------
ACTION
:    Vs EXISTS END
------------
   19 RECORDS SELECTED
------------
ACTION
:    Vs ( <= 2000 and >= 1000) END
------------
   11 RECORDS SELECTED
------------
ACTION
:    Sources (FIRST) CONTAINS "Mt. Wilson" END
------------
   1 RECORDS SELECTED
------------
ACTION
------------
:    DISPLAY SN Vs 12 b2 Sources (ALL) END
------------
SN       1901b
Vs       1617
12       271.15
b2       76.90
Sources
     1 Ap. J., 88 (1938), 285–304– Contr. Mt. Wilson 25 (1938)
     2 XIV Colloque Int. Astrophys., Paris (1941) 186.188
     3 Ann. Observ. Paris, 9, (1945) fasc. 1, 165–179, etc.
```

Although the above example only illustrates the simplest search capabilities of DIRAC-1, the command language also provided facilities for interactively creating files, updating them and displaying directory status. These facilities will be described below in more detail.

• Primary Design Implications

A primary design objective, whose implications are obvious in terms of user acceptance and training cycle, is the transparency of low-level modules. Non-procedural languages eliminate the need for users to maintain an awareness of physical file structure: a statement in the command language is broken into subsets that trigger the appropriate series of primitive calls. The resulting operations, in turn, are handled by the time-sharing system that provides the necessary link to the file system.

Placing file creation under user control has many immediate advantages; the major improvement is found in the fact that one no longer has to painfully explain the process of his work to a programmer, who may or may not be able to grasp in a few days the concepts of data organization (often specific to each discipline) that the specialist has acquired after many years of training. Another very important advantage is that users no longer have to *a priori* select one particular structure for their work. They can create a file with an experimental layout of fields and sub-fields, update it with a few typical records, attempt interrogation to see "how it feels," and repeat this cycle several times.

At this point it is important to emphasize a statement made by Bennett (*10*) concerning the role of the designer and his anticipation of a certain level of user expertise: "The user here is typically not professionally committed to the use of computer equipment per se." This statement was made in the context of library automation, but it is valid in the entire field under consideration here.

Early in 1970, when DIRAC-1 was made available to selected users at Stanford University, an important experiment was carried out to study the implications of the concept for scientists who used the system on-line. This experiment consisted in the creation and operation of a network prototype using data bases from astronomy.

In this experiment it was decided to study users who had little or no computer experience, did not wish to become computer experts, had a well-defined, pressing problem where computer support was mandatory, and could provide us with a large initial data-base. In addition, they had to be situated in a remote location (without access to the system designers). The users we thus defined exhibited the characteristics of a much larger class, namely, all those who would not use a procedural language.

Users with these interests and profile were found among the scientific staff of Dearborn Observatory, near Chicago. They were provided with a 2741 terminal and the ability to use DIRAC and a large astronomical data-base that was created at Stanford. The experiment lasted eight weeks, leading to the recording of 120 time-sharing sessions. At the end of the experiment, interviews were held with the participants to find out their attitudes towards the system and to analyze its implications in terms of their own work. We also used a simple technique of performance monitoring to record instances of system behavior of various types. These results have been reported in detail elsewhere (*3, 7*).

Typical of the questions asked in the course of this experiment was the problem: "LIST ALL IRREGULAR GALAXIES VISIBLE FROM EVANSTON IN JUNE FOR WHICH NO RADIAL VELOCITY HAS BEEN MEASURED." To solve such a problem by hand would require approximately one day of an astronomer's time, and to solve it by conventional programming could take approximately the same time, (assuming standard turnaround and a program with no bugs). This problem was solved in a few minutes using DIRAC-1. A primary implication was thus found in the fact that it became inexpensive to ask a type of question that would not have been justified under previous systems, and yet was most valuable to the users.

Such a system can be decomposed into six major components:

1. The data,
2. The machine itself,
3. The implementation language,
4. The medium,
5. The interface,
6. The problem area.

In the case of the astronomy network, the machine was given, the language was FORTRAN, the medium was magnetic disk storage, the interface was a remote terminal with no display capability, and the problem area was astronomical education and research. However, one could apply this model to an information system that did not contain a computer. The classical library, for instance, is a system where the "machine" is a matrix of shelves; the data are contained in bound volumes, keywords, nomenclatures and abbreviations; the medium is paper; and the interface is an index.

Is there an identifiable implementation language in the manually-operated library? According to Bennett (*10*) there is indeed such a language. It was developed to permit the interfiling of access points to the main collection and it consists of cataloging codes and filing rules.

Implications of design changes can be studied easily under this model. For instance, we can follow the procedure changes that would be triggered in the library by changing the medium (from paper to microfilm) or the machine (from material support to magnetic storage).

Primary design implications that we have identified can be stated simply as a set of guidelines for the designer of some future system.

1. It is important not to set up a rigid system. It must be possible to alter components without drastically re-designing operating procedures as seen by the end-user.
2. The user's responsibility must be identified very early to allow him to shape those parts of the system, notably the interface, that are critical for his application. (The procedure followed to effect this in DIRAC will be illustrated in the section "The User's Viewpoint.") The authors of the INFACT system, for instance, have stressed the point that the software was easier to develop when they proceeded "in an inductive direction, from the specific to the general" (*8*).
3. Users should never be locked inside the retrieval language. Multi-level interaction must be facilitated, rather than hampered, by the system. In particular it should be easy for the user to obtain information from the terminal-handler, from the operating system, from the file management utilities, and to link to compilers during search.
4. Implementation languages must be selected purely on the basis of ease of programming and inexpensive training.
5. The system should be as independent of specific data features as possible, and it should be the user's responsibility to provide the data definition in terms of allowable types.
6. Clearly-defined editing facilities should be provided. Data entry and all related functions are often underestimated in the early design of database systems. Such items usually prove much more expensive than initially thought, especially when they must include facilities for proofreading and correcting records already entered into the data-base.
7. Finally, the system should *not* be optimized for an existing language, but instead the implementation language should be allowed to evolve.

An adequate design strategy, in keeping with the above remarks, would take advantage of any convenient machine and would use the highest-level language available on that machine (without writing a new one). A first version of the language would then be offered, and it would be so open-ended as to permit inexpensive redesign in the course of application development if minor points required alterations. Real-life applications would be tried very early, and would lead to complete calibration of the system. A second, optimized version of the first system level would then be released. At the same time, the new concepts that may have emerged during the first series of applications would be gathered in a second level, used in expanding the applications, and so on. At each step in this "bootstrapping" one would make a serious attempt to recognize and extract crucial functions and to optimize in their terms, at the cost of inefficient code if necessary.

In spite of its usefulness, study of these primary design implications does not lead to a complete understanding of the complex impact of information systems. A better view at this level is obtained when one begins to analyze the common characteristics of the systems that have failed.

• Why Information Systems Fail

Information systems that have been developed in the past have failed to satisfy users for a large variety of reason, ranging from unreliability of the hardware to poor human engineering at the command language level. In addition to purely technical problems, the misunderstanding of data management plays a role in many failures.

These systems can be classified into major categories depending on the type of shortcoming as seen by the user, irrespective of cleverness of programming, development cost, or hardware ingenuity.

Many systems provide no data. Instances of this type of failure are found in cases where development has preceded understanding, and where a programming staff has begun its work on data management concepts without drawing inspiration from real-life applications. It is typical for such a project to become hopelessly entangled in its own software problems and to create difficulties in its own path. A new computer generation appears and forces re-design before any tangible results can be demonstrated.

Many systems provide useless data. In this class of system the entire development has usually taken place from the inside out, perhaps with some real "end-users" in mind, but with insufficient awareness of the type of decision-making such users experience. The information obtained from the system turns out to be trivial, overwhelmingly sophisticated, or both.

Many systems provide usable data . . . buried in noise. The comment is frequently heard in management circles: "We have a fine system, and we are fairly successful at keeping the computer and terminals operating, but every time a real-life question comes up, it takes longer to go through the output to locate the real paydirt than looking up the information myself."

Many systems provide relevant data . . . in the wrong form. Every category of professional users has its own way of handling, labeling and displaying information. The graphic techniques that are used daily by a geologist will puzzle an astronomer, and in the formulation used by biologists, the chemists may not be able to perceive relationships. Coding systems are an especially frustrating example of this: medical nomenclatures and the numbering schemes for libraries have very peculiar mathematical properties. They serve purposes that programmers cannot be expected to "guess at." For this reason, much of the information produced by current systems, relevant as it may be, turns out to be useless because it is not in a form users can recognize.

Our team had such an experience with the cancer research applications of DIRAC-1 during the transition phase to DIRAC-2. The results, that came on the terminal in the form of tables and numbers, had to be studied by the specialists and compared to previous data or results available in the literature. When this process was investigated we learned that cancer specialists had developed standard reporting procedures and standard measures of treatment effectiveness and survival rates. The system was then programmed to generate these statistics from the record subset selected by DIRAC. In addition it became possible to display the actual survival curves on the terminal. The pace of the medical analysis was clearly increased by this technique, which led to the identification of significant differences among subsets of patients that were previously not differentiated (*4*).

And this leaves all the systems that do provide relevant data, occasionally in a legible form, but *too late*. In this case the failure can usually be traced to poor integration of the system within the organization it supports; to poor training; to distrust on the part of those who generate and manage the information; or simply to software design that has created a cumbersome machinery that does more to *shield* end users from what is happening in their data-base than to provide decision-making power.

• Secondary Design Implications

Observation of these attempts at developing information systems leads to a number of remarks that may seem obvious but do require clear attention. These points have consistently been ignored in the design of the "bad" systems, in one way or another.

First, it is imperative to understand clearly the real nature of the user's problem. For example, the library environment, with its very special set of parameters, is not the ideal one in which to study the interaction of scientists. It is not true that the numbering scheme or the keyword mechanism used in a given profession will successfully serve the needs of another one. To some users it may be imperative to have access to the complete text of each document. In other applications an index or a cross-reference is adequate. An effort to force certain techniques, like the use of keywords, on those fields (such as medicine) where rapid evolution of the terminology is taking place, may lead to failure of an otherwise well-designed system.

Second, it is important to understand the dimensions and the goal of the application. Consider the case of astronomical research, where extremely massive catalogues have been accumulated and where the potential number of objects that could be classified has no upper limit. Careful observations of patterns of search show that a given astronomer does not need constant access to these catalogues, and that his work would be greatly simplified if a system allowed him to generate a "personal catalogue" of those objects he studies, with some form of on-line updating capability to draw upon the resources of the larger data-base. Supporting this kind of two-level information system will be more important than designing the fastest access mechanism for a single, centralized depository of data.

Third, the temptation to build "the ultimate information machine" should be resisted. The fact is that the state of the art in software does not provide us with enough knowledge to predict what functions future users will apply to data-bases. Instead, we should design very general tools with open-ended capabilities, and we should apply them to carefully-selected applications.

Reporting on his successful experience with such a general software tool, an MIT political scientist has called attention to this point: "No data management system is likely to solve in the near future all the problems of data analysis that the survey analyst now faces" (*11*).

Finally, interfaces should be kept clean and very simple. This, in itself, will eliminate several types of potential difficulties. It will shorten training time; clerical personnel with direct access to a lower-level system, such as a text-editor, will no longer have to be aware of the entire machinery behind it. It will make it easier to implement and maintain the system. It will make changes possible when a monolithic system would be vulnerable to alteration of its environment.

Drawing from these observations it has been possible to create a new version of the DIRAC language that is as simple in its command structure as DIRAC-1 but has more powerful internal search and display capabilities. The following are some of its features:

1. It is fully interactive and operates in a time-sharing environment.
2. It appears to the user as a natural extension of a text-editor with which it is interfaced.
3. Data entry is effected remotely and only involves simple typing skills.
4. The system introduces an order-of-magnitude reduction in the cost and delay of retrieval operations compared to classical computer programming.
5. The commands may easily be learned in one session at the computer terminal of less than an hour's duration. No previous programming knowledge is required.
6. Responses to the most important search commands may appear either on the typewriter terminal or on a video display scope at the user's request.
7. Selected records from the main file can be dynamically designated and treated as separate sets during later search phases.
8. It allows a combination of indexing (user-supplied codes such as industry standards, government designations, abbreviations, etc,) with user-specified field designations in free format.
9. The self-defining structure of DIRAC-2 makes updating especially easy. Furthermore all text-editing commands are recognized for modifying data in any way.
10. Statistical tools are provided in the form of correlation matrices and histograms that can be triggered and displayed through simple commands that apply to any file in the system.

The geoscience data-base that we currently use contains information of potential interest to three main categories of users. The first user, the government administrator faced with decisions on alternate uses of a particular region of public land, may wish to assess its mineral resource potential. Typical of the second user class would be the personnel in charge of exploration for a large mining company. Finally, a geologist might be interested in correlations among certain mineral assemblages found in particular deposits.

It is useful to ask how they might each take advantage of a computer technique such as we have illustrated, a question that the literature of geoscience applications has begun to examine (*14*). The mineral resource potential of the land area studied by our first user (the government administrator) will be accessible to him through correlation and histogram displays; our second user (the exploration man) will require more detailed information and will apply a variety of search commands to the data base until he has selected subsets that satisfy his criteria for the commodities with which he is concerned. Once this search is completed the system will give him access to the on-line bibliography for further reference. The third user, our geologist engaged in research, will combine both of these approaches until his questions are refined into a consistent problem, to which will correspond a precise set of data records.

• From DIRAC-1 to DIRAC-2: A User's Viewpoint

The reader will find in the Appendix an actual terminal listing of a DIRAC-1 session aimed at defining a mineral resource data-base. This session took place early in 1971 at an IBM 2741 terminal and utilized the Stanford IBM 360/67 computer to create a prototype file for mining properties in Montana. The aim of this experiment was not to create a permanent file, but to acquire a sufficient grasp of the problem of structuring such a data-base to make recommendations regarding larger problems, such as we shall see in the next section.

DIRAC-1 was so designed that questions or "prompts" were given to the user, who then responded with answers. In some instances only certain responses were recognized (such as the choice of output device) whereas in other instances the user had complete freedom in entering terminology of his own choosing, such as the selection of names and descriptions of the various fields. When an invalid response was made, the user was informed and supplied with the range of appropriate replies.

In the example given in the Appendix, which illustrates the self-defining features of the language, queries generated by the computer are flush with the left margin, are always capitalized, and usually are preceded by a number. The prompt to the user is a colon, as in the astronomical example seen above, and a combination of upper and lower case characters is permitted. Explanations of what transpires during this session refer in most cases to the actual paragraph numbers preceding the prompts; in some cases, letters (A, B etc.) have been supplied.

A The user specifies that a file is to be created.

B The user's own file identification is provided.

C Simple prompts will be used under the "terse" mode, as opposed to a more tutorial, or "verbose", mode of operation.

1 The file will also be known by the name "minerals".

2 The string "***" indicates the place where the user recalled that the description must be enclosed in double quotes. By striking the ATTN key he was able to correct his error.

3 This specifies who can use the file: in this instance, anyone can. However, only two persons are named as permitted to update (add to, modify or delete) the file.

4 A special notation could be defined at this point.

5 This length of 512 characters is based on an estimate of the maximum length anticipated for any one record. It can be changed on the basis of update statistics. Here the dynamic revision capability of DIRAC-1 was used to change the preliminary estimate to 256 words.

6 "Fields" refer to the various classes of data that are to be entered. These again are user-selected.

7 The user is requested to indicate if the various fields will contain single or multiple entries.

8 The record locator is standard.

9 No structure specification beyond the tree structure already defined.

10 Validations are imposed on the file by the user. For instance, no record will be accepted that has over fifteen entries for economic elements.

D The string *********! is an error feedback signal that points to the last position scanned. The user then receives an error number and information regarding the acceptable specifications at this point of the "conversation." This response is offset to the right to preserve readability, and the system recovers normally.

E At this point the structure of the file is completely defined.

F Another case of error and correction on the user's part.

G The user is requested to specify the mode of display of the status report.

H This report indicates the organization of the file, as well as other relevant information; statistics will be entered dynamically as updates take place. This report is automatically formatted by DIRAC.

I We now enter update mode.

J New records are to be entered from the terminal.

K Entries are made for the various fields in the first record. It is noteworthy that fields can be specified in any order, and that the entire update session could also take place either from a previously created 'update set' or from a formatted tape, under the positional input processor of DIRAC.

L The second record is now created.

M We now begin interrogation to check the feasibility of working under this preliminary structure.

N Again, an improper response is made, and the correct form is provided.

O The request is made to select those records (deposits) that contain gold.

P Both records satisfy this condition.

Q What are the gangue minerals that are known to occur in these deposits?

R The next request is made to select those deposits which contain both gold and lead.

S Only one property satisfies this condition, and we type out all the information concerning it.

T The total time involved in this entire experiment, from file creation to interrogating a sample with two complete properties, was under thirty minutes.

Development of the application beyond the stage of "academic" research led to major decisions in terms of system support and language structure when the problem of automating the large data-base of Alaska mineral resources was presented, as the next section will show.

• The Alaska Data-Base: Research in Progress

The largest systematically organized file on mineral deposits for any one state is that of Alaska, which was initiated in 1948 and represents approximately ten man-years of work.

The heart of this file consists of approximately 7,500 mineral occurrence locations recorded on five-by-eight cards stored in Kardex cabinets. These in turn are tied to a master bibliography of all Federal and State mineral resource publications which describe these occurrences.

The basic breakdown of this file is by USGS quadrangle-maps at a scale of 1:250,000 of which there are 153 in Alaska with about two thirds of these reporting occurrences of metallic mineral deposits.

A typical example of the kind of data recorded and the manner in which it is presented is shown below.

This file, in its existing form, has served as the basis for over 75 publications and maps, the source of answers to numerous requests for information on Alaska, and over 2500 copies of mineral occurrence reports with accompanying bibliographies and maps have been distributed to individuals and firms upon their request.

```
(Fortyfive Gulch) (Pup) (Pass)       GOLD, TIN, TUNGSTEN
Ellsworth and Parker, 1911,          64d12m N. 142d05m W approx
  p. 170                             Eagle (14.5, 3.6) approx.
Porter, 1912, p. 217
Ellsworth and Davenport, 1913
Mertie, 1938, p. 185
Joesting, 1943, p. 19–20, 28
Thorne and others, 1948, p. 28
Smith, W. H., 1968 p. 2
Foster, 1969a, p. G28
Foster and Clark, 1970, p. M5
(Fortyfive Gulch) (Pup) (Pass)          Au, Sn, W
```

In 1971 we began experimenting with DIRAC-2 to ascertain the feasibility of automating this file without significant modification of use patterns, but in a way that would considerably extend its scope. First, a trial file of fifty-one properties was created with selections from various quadrangles. The following sessions illustrate a typical query sequence in the abbreviated form of DIRAC-2.

In the first part of the session, a request is made to find all those properties in which copper is reported. This subfile is retained for interrogation or display at a later time. Next, we learn how many of the occurrences indicate lead and silver mineralization and are located in the Nulato quadrangle. We find that two such properties exist. We type out all the information known about the first of these two sites. Next we wish to obtain the complete bibliographic reference for the article by Cass. The master bibliography is searched, and the reference is typed out.

```
? find ELEM Cu
   15 ENTRIES
? hold 1
? file 0
   51 ENTRIES
? find ELEM Pb
   9 ENTRIES
? and ELEM Ag
   2 ENTRIES
? hold 2
? and QUAD NULA
   2 ENTRIES
? type
NULA Perseverance
        <64*30'N, 157*09W>
        Brooks, 1923, p. 39
        Brooks and Capps, 1924
        Mertie, 1936, p. 227
        Cass, 1959
        Berg and Cobb, 1967, p. 228
? list 'Cass 1959'
    Cass, 1959—Cass, J. T., 1959, Reconnaissance Geologic map
of the Nulato quadrangle, Alaska: U.S. Geol. Survey Misc. Geol.
Inv. Map I-291, scale 1:250,000,
```

Returning now to subfile 1, it is desired to examine the number of times copper is found to occur in connection with other economic elements. A correlation display demonstrates this immediately. We also produce a histogram to indicate the total and relative occurrence of each of these economic elements in those fifteen copper-bearing occurrences. This information would be extremely difficult and costly to obtain if we had to rely on a manual search on the entire file.

```
? file 1
   15 ENTRIES
? corr ELEM ELEM
? crt 0
                    ELEM                    ELEM
                           ! Sb ! Cu ! Au ! Pb ! Ni ! Ag !
             -----------------------------------------------
19 18.1%     ! Sb    ! 5  ! 5  ! 5  ! 4  ! 0  ! 0  !
35 33.3%     ! Cu    ! 5  ! 15 ! 7  ! 5  ! 2  ! 1  !
25 23.8%     ! Au    ! 5  ! 7  ! 7  ! 5  ! 0  ! 1  !
19 18.1%     ! Pb    ! 4  ! 5  ! 5  ! 5  ! 0  ! 0  !
 4  3.8%     ! Ni    ! 0  ! 2  ! 0  ! 0  ! 2  ! 0  !
 3  2.9%     ! Ag    ! 0  ! 1  ! 1  ! 0  ! 0  ! 1  !
             -----------------------------------------------
TOTAL = 105            19    35    25    19    4     3
                       18.1  33.3  23.8  18.1  3.8   2.9
             -----------------------------------------------
? draw 0
       ELEM
 5 14.3% Sb   *****
15 42.9% Cu   ***************
 7 20.0% Au   *******
 5 14.3% Pb   *****
 2  5.7% Ni   **
 1  2.9% Ag   *
```

Among the advantages of automating this file under DIRAC-2 we found the following:

1. The data-base can remain at one location where it can be maintained, yet may be accessed instantly from any point where terminal service is available.
2. People interested in *selective* information—such as all Cu-Ni occurrences—can obtain instantly all known reported locations and an accompanying bibliography, previously a very time-consuming task.
3. It is possible to keep this information source current, by updating both the literature references and the parameter values stored in the record, whereas published material becomes rapidly obsolete.
4. One may seek out relationships which could prove of value, but which would be too time consuming and costly to obtain manually.
5. The system meets the short-term response requirement of decision-makers and administrators.
6. The data-base can be interfaced with plotting routines to produce commodity maps, such as those created for S.E. Alaska by Heiner and Wolff in 1970.

Current work is proceeding in line with the requirements, recently set forth by the U.S. Geological Survey, to use computerized data banks for resource information *(12)*.

• Principles of "Datanautics"

How could we apply the knowledge of these implications to the design of better information systems?

A special item of interest here is the generalization of the concept of a set of files into that of an information space. In the library context, Bennett poses the following questions: "Is a sense of spatial location important to a searcher? Are there display techniques that give the searcher a sense of continuity as he skips from place to place within a system file? Between system files?" *(10)*.

These questions have been addressed by several designers. The Augmentation Research Center at SRI has proposed and demonstrated ways of linking documents within on-line files. Given that techniques do exist to move at will within such an information space, what kind of navigational aids do we need?

It will be useful, as the first step, to clearly distinguish between the automaton, or system, that performs the text manipulation functions, and that system which communicates with the user at the highest level during search. Given the reference plane of the user, and given the various layers of information that are visible from this plane, we can imagine that the first type of software tool at his disposal might serve the purpose of a "data probe" with the following functions:

1. COMMUNICATION. In this mode messages can be sent and received at the terminal, "help" commands are processed and error messages are posted.
2. OBSERVATION. The language at this level has the ability to describe the structure within which the user is moving.
3. MOBILITY. The user needs the ability to move across files.
4. ACQUISITION. Ability to acquire the record specified by an index, to select a part of text, etc.
5. VISUALIZATION. Examples of this function might be to move the third line to the top of the screen, to display selectively the context of certain keywords, etc.
6. ORGANIZATION. All the functions of editing at the word or character level.
7. DERIVATION. The ability to operate on the file and generate (compute) new items.
8. TERRITORY. The user must be provided with a sense of the boundaries within which he moves, including any privacy concepts that may be applicable.

Such a "probe" may be a text-editor with terminal handling capabilities (although such programs typically provide little computing power). It is important to specify the type of visual output available at this level both in terms of the data itself and of the structure of the data. Traditional design principles are applicable here. A classical book on human engineering *(1)* has this to say about the display of motion within a structured space:

In general it can be said that, as an operator's vehicle moves about in space, he can best comprehend a display that gives a representation of his movement with the fixed portions of the instrument representing the space within which he has moved.

This principle can be applied to the allocation of display areas on a scope for the visualization of the data structure as opposed to text, and it has a place in a discussion of human factors.

The scond software level that will communicate with the probe through a simple interface could be called "information control module" and would have the following functions:

1. DATA MANIPULATION. All functions of the "data probe".
2. HIGH-LEVEL TASKS. File system functions are transparent.
3. INVULNERABILITY. Recovery from failure or conflict.
4. MACHINE INDEPENDENCE. Hardware and operating system functions are transparent.
5. AUTOMATISM. Non-procedurality of the command language.
6. ANALYZABLE. Gathers information about itself and monitors its own performance.

When implementing such a system, one should identify these components and anticipate a processor with an analysis level and a recovery level for each component.

Some of the above points may appear as philosophical issues of little significance to the "nuts and bolts" implementation work, until one finds himself faced with the real-life complexities of natural language processing. In the context-sensitive environment of the newer systems, the detection of ambiguity places an additional burden on the control modules.

In an experimental system developed in 1966 for question-answering in astronomy, the problem arose of distinguishing between such questions as:

FIND THE PERCENTAGE OF GIANTS BLUER THAN A2 AMONG BINARY STARS WITH VELOCITY BETWEEN 50 AND 70 KM/SEC;

that would lead to feasible retrieval formulae (and a numerical answer) as opposed to questions of the type:

HOW MANY STELLAR SYSTEMS HAVE STARS?

where the answer could be determined without search of the data-base, while a third type of question, such as:

HOW MANY MULTIPLE STARS ARE SINGLE?

or:

WHAT IS THE PROPORTION OF DWARFS AMONG GIANTS?

resulted in semantically-invalid interrogations. Clearly the automaton that determines semantic contents of such queries should be separated from the module that will, for instance, recognize DWARFS as a plural form of the word DWARF.

• Questions about Information

After considering the questions we have enumerated, we must ask whether some very basic difficulties, inherent in the nature of information processes, will not place severe limitations on the best software system we can ever hope to design, given our present understanding of these processes. These 'theoretical limits' are already reached in some instances, although programmers are proposing new ways to get around some of these difficulties.

Consider as an example the case of a hospital data-base that has three potential categories of users:

The first category of users, *hospital administrators,* may wish to ask questions such as: "How long do people stay in ward 17?" or "How many beds were available in June?"

In contrast to this type of question, the *clinical staff* will query the same system to obtain data concerning tests for particular patients, or the number of sleeping pills to be distributed this evening.

A third category of potential users, the *medical researchers,* may query the system for long-term correlations between complications and symptoms in given diseases.

What are the characteristics of an information system that would serve all three categories of users in a high-level language, and in optimal fashion? Consideration of this problem immediately leads to basic questions about the nature of information, and raises the issue of the definition of languages to describe its structure. Some questions that we must pose at this level follow.

Is information structure "God-given?" In other words, is there, for any problem, an optimal structure specification that must be discovered (or somehow "revealed") at the outset, and never altered? If so, for a given problem, can we build a set of "primitives" that represent the structure, and can we deal entirely with these primitives in later stages? Such an assumption appears to underlie most of the work currently done in library automation, yet the hypothesis has not been examined in the light of the difficulties we have alluded to.

Even more basically, designers could look into the problem of determining whether it is always possible to separate structure from data: in the description of astronomical entities, such as stellar systems, we already found that this was not necessarily the case. In a multiple stellar system, for instance, the word "triple" is an answer to a data question ('What is the multiplicity of the star Castor?') but it also determines the number of branches we see in the information tree at the next level.

In a system where structure manipulation is inexpensive, the user himself can rapidly evaluate the implications of structural changes. In other words, the system can be used not only to implement a particular structure, but to *discover* the "best" structure among several alternatives.

As experience with information retrieval spreads outside the scope of the Library, and reaches many levels of government, business and research, it is becoming increasingly apparent that other major difficulties have not yet been identified.

Computerized information retrieval is a difficult field, because it is not usually possible to store "information" in a computer in the first place. All we can store in a computer is *data.* These pieces of data, when retrieved by a human user, may appear as information to that particular user, but it is unrealistic to expect the parameters of this interaction to be measurable in the terms of transactions observable at the machine interface. Such an interaction can only be evaluated in terms of the behavior of the total organization, not on the basis of CPU cycles, number of terminal transactions or disk utilization ratios.

Given a user who is trying to manipulate research data, an effective data-base system must function as a re-structuring tool rather than as a simple processing machine. It must adapt to intellectual decisions made by a human and it must be capable of displaying information at varying depths and in a great variety of forms, not so much as a support for retrieval of a particular document but rather as a guide to the user's thinking process. A crucial, yet seldom recognized, decision that must be made early in the development of an advanced system is the selection of a significant application to serve as a guide for the implementer's thinking. A person analyzing this kind of interaction might find more inspiration in the work of artists, theoreticians in science or decision-makers in business, than in the observation of computer programmers or librarians.

In this context, it is not enough to observe the representation (the finished work of art, the demonstration of a theorem, or the form of an investment decision). We must ask what was the intent of the activity, its meaning in a particular context, and we need to identify the components of this interaction. Then we need to discover how the representation or activity has evolved during the history of this particular user, or school of users, before we can form a theory of the process and measure its effectiveness. This is a problem of much greater scope than currently defined.

As we begin to question the traditional view of the user as a passive recipient of information, it is not enough to recognize his role as a source of questions. In an information system the human plays a far more important role, one that has so far been regarded as lying outside the scope of system design: his role as an operator in the mathematical sense, as the originator of transformations that can be applied to the data. The result of the transformation is generally what is recognizable, and retrievable, *not the data itself.* Thus we are led to the speculation that future data-base systems will have to devote increasing attention to the support of a class of intellectual processes that operate upon complex information structures.

References

1. Woodson, W., *Human Engineering Guide for Equipment Designers*, University of California Press, Berkeley 1960.
2. Montgomery, C. and R. Katter, personal communication.
3. Vallee, J., et al., "The Organization of Research Data Banks," *Proceedings of ASIS* 1971 pp. 387–394.
4. Ray, G., J. Cassady and M. Bagshaw, "Prostatic Carcinoma Treated Definitively by External Megavoltage Irradiation: 14 Years Experience," in: *Radiology*, forthcoming.
5. Wolf, P., J. Vallee, et al., "Progress towards a Direct-Access Hematology Data-Base: Stanford's Experience with the DIRAC Language," *Archives of Pathology*, 91: pp. 542–549 (June 1971).
6. Englebart, D., and Staff, *Advanced Intellect-Augmentation Techniques*, NASA Report, SRI, July 1970.
7. Vallee, J. and J. A. Hynek, "DIRAC and Astronomical Data Retrieval," in *Computers and Crisis*, ACM 1970 National Convention, New York, Sep. 1970.
8. Rubin, S. and D. O'Connell, "A Self-defining Data Structure," ACM-SICFIDET Workshop on Data Description and Access, Rice University, Nov. 1970.
9. McIntosh, S. and D. Griffel, "The current ADMINS system for non-textual data," Center for International Studies, MIT, Dec. 1967.
10. Bennett, J., "Interactive Bibliographic Search As a Challenge to Interface Design" in *Interactive Bibliographic Search* (D. Walker editor), AFIPS Press. Montvale, N.J. 1971.
11. Silva Michelena, J., "ADMINS—A User's point of view," MIT report, 1967.
12. "Mineral Resource Appraisal Program," U.S. Geological Survey, 1972. (Internal document, peronal communication)
13. Askevold, G., "Information Systems and the Management of Mineral Resources," Ph.D. Dissertation, Department of Applied Earth Sciences. Stanford University. (forthcoming).
14. Hruska, J. and C. F. Burk, "Computer-based storage and retrieval of geoscience information: bibliography 1946–69" Published by the Geological Survey of Canada. GSC Paper 71-40. Department of Energy, Mines and Resources, Ottawa 1971.

Appendix
A Complete DIRAC-1 Session

```
?
enter
                         DIRAC VERSION 1B
NAME OF USER
  : Askevold
PLEASE TYPE EXECUTION MODE
  : CREATE                                 (A)
FILE IDENTIFICATION
  : G010                              (B)
KEY FOR THIS MODE
  :
COMMAND
  : SET TERSE                         (C)
INTERFACE
  : NONE
------------
 1. FILE NAME
    : Minerals
 2. FILE "DESCRIPTION"
    : File of ***
    : "File of Mineral Deposits"
 3. DISPOSITION (PUBLIC/PRIVATE)
    : PUBLIC
      ENTER "AUTHORIZED UPDATE USERS"
    : "Askevold Other name"
      --------------
      Revisions ?
    : NONE
 4. SPECIAL NOTATION FOR RECORD AND FIELD?
    : NONE
 5. RECORD LENGTH?
    : 512
------------
      REVISIONS?
    : REVISE 5
------------
 5. RECORD LENGTH?
    : 256
      ---------------
      REVISIONS?
    : NONE
 6. SUPPLY NAME AND "DESCRIPTION" FOR ALL THE FIELDS
@1  ? Property "Name of Property"
@2  ? District "Mining District"
@3  ? County "County"
@4  ? Survey "Land Survey"
@5  ? Type "Type of Deposit"
@6  ? Elements "Economic Elements"
@7  ? Production "Past Production"
@8  ? Ore "Ore Minerals"
@9  ? Gangue "Gangue Material"
@10 ? Host "Host Rock"
@11 ? Alteration "Alteration"
@12 ? END
      REVISIONS ?
    : NONE
 7. SUPPLY DATA TYPE AND MULTIPLICITY
    : ALPHA SINGLE @1 @2 @3 @5 @11 @4
    : ALPHA MULTIPLE @7 @10 @9 @6 @8
 8. DEFINE "RECORD LOCATOR"
    : NONE
      ---------------
      REVISIONS?
    : NONE
 9. DEFINE RECORD STRUCTURE
    : NONE
10. VALIDATION
    : @6 SUBFIELDS 15
    : @7 SUBFIELDS 20
    : @6 LENGTH 30
      *********!
------------ --------                 76          (D)
                                       ACCEPTABLE SPECIFICATIONS:
                                       NECESSARY, SIZE, SUBFIELDS,
                                       FOLLOWED BY INTEGER.
    : @6 SIZE 30
    : @1 NECESSARY
    : END
      -----------------
      REVISIONS ?
    : NONE
```

```
11. CODES: NONE
-----------
THE FILE HAS BEEN CREATED.                    (E)
  :  SET VERBOSE                              (F)
     ***!
---------------1
```

```
VALID COMMANDS: CREATE, QUERY,
UPDATE OR STATUS, UPPER CASE.
  :  STATUS G010
OUTPUT DEVICE? (0/1/2 = TERMINAL/SCOPE/EDITOR)     (G)
  :  0                                             (H)
```

```
------------------------------------------------------------------------------------------------
STANFORD UNIVERSITY                    STATUS REPORT FOR FILE G010                     DIRAC-1
COMPUTATION CENTER                            16/FEB/1971
                                  DESCRIPTION: File of Mineral Deposits
CREATION          16/FEB/1971                                       FILE NAME  Minerals
RECORD            $N$                                              CREATED BY  Askevold
FIELD             @F                                            RECORD LENGTH  256
DISPOSITION       PUBLIC                                        NO. OF FIELDS  11
NO. OF RECORDS    0                                             LATEST UPDATE  16/FEB/1971
------------------------------------------------------------------------------------------------
FIELD IDENTIFICATION, STATISTICS AND VALIDATION INFORMATION
------------------------------------------------------------------------------------------------
                                                TYPE          VALIDATIONS          STATISTICS
     FLD NAME          DESCRIPTION               TM     NEC      l         m        REC      PCT
------------------------------------------------------------------------------------------------
      1 Property       Name of Property          AS     YES      0         0         0        0%
      2 District       Mining District           AS              0         0         0        0%
      3 County         County                    AS              0         0         0        0%
      4 Survey         Land Survey               AS              0         0         0        0%
      5 Type           Type of Deposit           AS              0         0         0        0%
      6 Elements       Economic Elements         AM             30        15         0        0%
      7 Production     Past Production           AM              0        20         0        0%
      8 Ore            Ore Minerals              AM              0         0         0        0%
      9 Gangue         Gangue Materials          AM              0         0         0        0%
     10 Host           Host Rock                 AM              0         0         0        0%
     11 Alteration     Alteration                AS              0         0         0        0%
------------------------------------------------------------------------------------------------
```

```
  :  UPDATE G010                              (I)
FIXED FORMAT ?
  :  NONE
INTERFACE
  :  NONE
NEW OR OLD ?                                  (J)
  :  NEW
-----------
RECORD NO. 1
-----------
  :  @1 "Cape Nome"
  :  @2 "Clinton-Potomic"
  :  @3 "Missoula"
  :  @4 "SE 1/4, Sec. 19, T.12N, R.16W"       (K)
  :  @5 Vein
  :  @6(1) Cu
  :  @6(2)  Ag
  :  @6(3)  Au
  :  @7(1) ">19,000 lb"
  :  @7(2) ">2,000 oz"
  :  @7(3) "3 oz"
  :  @8(1) Chalcopyrite
  :  @8(2) Tetrahedrite
  :  @8(3) Malachite
  :  @9(1) Quartz
  :  @9(2) Barite
  :  @9(3) "Specular Hematite"
  :  @9(4) Siderite
END
-----------
RECORD NO. 2
-----------
  :  @1  "Legal Tender"
  :  @2 "Alhambra-Warm Springs"
  :  @3 Jefferson
  :  @4 "Sec. 9 & 10, T.8N, R.3W"             (L)
  :  @5 Vein
  :  @6(1) Au
  :  @6(2) Ag
  :  @6(3) Cu
  :  @6(5) Zn
  :  @6(4) Pb
```

```
  :  @8(1) Galena
  :  @8(2) Tetrahedrite
  :  @9(1) Quartz
  :  @9(2) "Manganese Oxide"
  :  END
MAXIMUM RECORD LENGTH : 104 WORDS.
THE FILE CONTAINS 2 RECORDS.
UPDATE EXECUTION TERMINATED.
  :  QUERY G010                                        (M)
COMMAND
  :  @6 CONTAINS Au END                                (N)
     **!                       -----------50
                               PROPER RESPONSE: SET TERSE,
                               SET VERBOSE, NONE
COMMAND
  :  SET VERBOSE
INTERFACE
  :  NONE
-----------
ACTION
  :  SELECT
  :  @6 CONTAINS Au END                                (O)
-----------------------------
2 RECORDS SELECTED                                     (P)
-----------
ACTION
  :  LIST @9
     ****!                    -------------17
                              SELECTION RULES BEGIN WITH @F,
                              SELECT OR LEFT PARENTHESIS
-----------
ACTION
  :  TYPE @9                                           (Q)
-----------------------------
 1
Gangue        1 Quartz
              2 Barite
              3 Specular Hematite
              4 Siderite
-----------------------------
 2
Gangue        1 Quartz
              2 Manganese Oxide
-----------
```

```
ACTION
  :  Elements CONTAINS  Au
  :  AND Elements CONTAINS Pb END                    (R)
--------------------------------
  1  RECORD SELECTED                                 (S)
------------
ACTION
  :  TYPE ALL
------------------------------
2
Property                  Legal Tender
District                  Alhambra—Warm Springs
County                    Jefferson
Survey                    Sec. 9 & 10, T.8N, R.3W
Type                      Vein
Elements                  1 Au
                          2 Ag
                          3 Cu
                          4 Pb
                          5 Zn
Ore                       1 Galena
                          2 Tetrahedrite
Gangue                    1 Quartz
                          2 Manganese Oxide
------------
ACTION
  :  END
NOW YOU CAN EXIT BY TYPING AN EXCLAMATION MARK
EXECUTION MODE ?
  :  !
EXIT FROM DIRAC-1
? logoff
EDITING TIME      = 2.79 SECONDS                (T)
COMPUTE TIME      = 15.68 SECONDS
MEMORY USAGE      = 799.83 PAGE-SECONDS
I/O ACTIVITY      = 0 UNITS
ELAPSED TIME      = 00:28:38
END OF SESSION
--------------------------------------------------------------------------------
                                                         9 June 1972
```