METHODOLOGY FOR COMPARATIVE SELECTION OF INTERACTIVE DATABASE INTERFACE TYPES

LENYA K. CRISTIANO

ABSTRACT

On-line database applications are becoming the most common new software tasks. Their use is becoming increasingly popular in all areas of information management. In many environments these on-line applications are made available to a large, diverse user population. The majority of these users do not have training in database or software areas. For this reason, the interface between the user and the database is vital. It serves to protect the integrity of the database by governing user access and it provides an understandable medium through which the untrained user can obtain or update the desired data.

The appropriateness of the interface can affect the success or failure of the database application. The wrong type of interface can result in inefficient data access and a reluctance on the part of users to utilize the tool. Most interactive database interfaces fall into two general categories, menu-driven or command- driven. These two interface types are dissimilar in appearance, usage, and often in performance. This article will provide a set of generalized criteria to assist the database manager or designer in selecting which interface type is better suited for a given application. The same criteria can also be used by managers in selecting a commercial database product for purchase. Factors such as the logical structure of the data, characteristics of the user community, and cost are considered. A worksheet is provided to allow for a quantitative analysis of the criteria in order to establish a foundation for the decision-making process.

Introduction

On-line database applications make up a significant part of today's business and information processing environments. These applications often represent a sizeable investment, whether in terms of development or purchase costs. Unlike the steady decrease in hardware costs relative to total system cost, the associated people costs have become one of the most significant system cost factors. While early computer users were largely dedicated users with a technical background, modern day users include grocery clerks, bank tellers, secretaries, and managers. These two trends - the comparatively high personnel costs and the broadening of the user population - have combined to place a much greater importance on the ease with which computer systems can be learned [Thomas(1)] and the efficiency with which they can be incorporated into a user's existing work environment.

It is vital that the product provide the maximum level of productivity possible, both for the software personnel who must build and maintain the system and for the end users who must utilize it to perform their jobs. To accomplish this goal the database components, such as the physical data file structure, I/O processing technique, and user interface must each be carefully selected to best fit the specific needs of a given application.

1.0 Interface Role

The design of the user interface provides the greatest opportunity for flexibility in generating a customized product. The user interface serves a critical and varied role in the modern database environment. The basic technical task of the interface is to serve as a buffer between the human user and the database management system (DBMS) which controls the stored data files. For this discussion, the term interface will be used to indicate the total software package surrounding the DBMS, including embedded DBMS commands, data translation required between DBMS and user formats, input data validation, and the interactive terminal package or visual screen displays.

The interface provides the user with a way to access information contained in the database. It then serves as a translator between the user and the DBMS to control input and output of data in formats that are understandable by

each. The interface also serves to guide the user through tasks by providing operating instructions, self-help information and error messages.

The interface screen displays make up the visible portion of the application. It is the most recognizable component and serves to identify the application to the user. The appearance or "style" of the interface can be important in itself. The interface design establishes an image for the product. The amount and format of displayed data as well as the presence of non-data features such as logos or other graphics and user instructions serve to alert the user to the nature of the application.

It may be said, in general, that the primary purpose of the interface is to serve the needs of the user while the needs of the data are handled by the DBMS. However, the logical structure of the data should also serve to influence some aspects of the interface. The format and amount of space required to display a single logical, coherent unit of data should be considered. For example, an interface designed to display small, independent data items (such as numeric data) would be ill-suited to work with data organized in block or page format (such as text).

Users should be able to access specific data quickly and easily. The interface design should strive to eliminate the display of any extraneous data which is not of immediate relevance to the user's query. The query syntax should be simple and straight-forward and should be consistent throughout the application.

A poorly designed or incorrectly selected interface can prove to be a serious deterrent to the success of an application. An interface which is installed and presented to the end users without review or consideration of their unique requirements is likely to result in inefficient data access and productivity levels well below expectations, at best. At worst, users may become frustrated with the system and so avoid using it. The adverse effects of such a poor decision can linger long after the initial error as users may tend to view later projects with preconceived attitudes or predictions of failure, regardless of the merits of those projects.

2.0 Interface Types

Interfaces are as varied as the applications they are built for. Most are based on one of two fundamental designs, command-driven or menu-driven. The visual differences between the two types makes them easy to distinguish. Other significant differences equip each interface type to best serve a given data file structure and user community.

Menu-driven interface systems consist of a number of individual, full-screen displays (refer to Fig. 1). The displays are commonly organized in a hierarchical manner. The user may choose to follow any of a number of prescribed paths or branches emanating from the primary or root screen, often referred to as the primary menu. The user moves from one screen or page to another by selecting from the options available on the current screen. The lay-out or format of each screen is constant, only the data used to fill the screen display fields varies. In more advanced systems, windowing may be used to combine screens for simultaneous viewing.

Command-driven interfaces utilize a less structured format. User commands and database responses follow a straight-line, scrolling pattern (refer to Fig. 2). This type of interface is best suited for less complex input and output forms. In its true form, the attraction of this interface

Figure 1. Menu Driven Interface

```
*********************************************************
*                ACME MAIL-ORDER, INC.                  *
*                                                       *
*                                                       *
*                     MAIN MENU                         *
*                                                       *
*                1 - Customers                          *
*                                                       *
*                2 - Order Forms                        *
*                                                       *
*                3 - Customer Accounts                  *
*                                                       *
*                4 - Inventory                          *
*                                                       *
*********************************************************
*             99 - Help         0 - Exit                *
*********************************************************


*********************************************************
*                ACME MAIL-ORDER, INC.                  *
*                                                       *
*                                                       *
*          Customer # : J00803                          *
*                                                       *
*               Name : MARTIN, JAMES M.                 *
*                                                       *
*             Address : 503 QUAIL RUN LANE              *
*                                                       *
*     City, State, Zip : BOSTON, MA 20387               *
*                                                       *
*               Phone : (451) 665-7301                  *
*                                                       *
*********************************************************
* 1-Add Item   3-Modify Item   7-Delete Item   0-Exit   *
*********************************************************


*********************************************************
*                ACME MAIL-ORDER, INC.                  *
*                                                       *
*  Customer #: C40023                                   *
*                                                       *
*  Select   Date     Item #      Qty     Unit Price     *
*                                                       *
*    *     3/25/88  100-1A        12    $    39.95      *
*    *     3/25/88  200-2A         5    $    12.50      *
*    *     3/25/88  K1011          2    $   120.00      *
*    *     5/10/88  3046-B         1    $    29.95      *
*    *    **/**/**  ********    ****    $******.**      *
*    *    **/**/**  ********    ****    $******.**      *
*********************************************************
*   1-Add Items   3-Modify Selected Items   0-Exit      *
*   2-Next Page   7-Delete Selected Items               *
*********************************************************


*********************************************************
*                ACME MAIL-ORDER, INC.                  *
*                                                       *
*  Customer #: K00917                                   *
*                                                       *
*                      PAYMENT                   1 cash *
*                                                2 chck *
*  Date: **/**/**   Amt: $******.**    Code: *   3 visa *
*                                                4 mstc *
*********************************************************
*     Curr          Amt       Date         Account      *
*    Balance        Due       Due          Status       *
*                                                       *
*  $  635.12    $  20.00    9/15/88       current       *
*********************************************************
*    1-Record Payment    3-Modify Record     0-Exit     *
*********************************************************


*********************************************************
*                ACME MAIL-ORDER, INC.                  *
*                                                       *
*                  Inventory Listing                    *
*                                                       *
*  Select  Item#     Descrip      Unit Price   In Stock  *
*                                                       *
*    *     100-1A   Crystal Vase    $  39.95      32     *
*    *     200-2A   Frame, 5x7      $  12.50      53     *
*    *     23684    Fire Screen     $  55.00      12     *
*    *     3046-B   Elec Skillet    $  29.95      44     *
*    *     H776-3   TV, 19" BW      $  89.95      21     *
*    *     K1011    Luggage, 3pc    $ 120.00       8     *
*********************************************************
*   1-Add Items   3-Modify Selected Items   0-Exit      *
*   2-Next Page   7-Delete Selected Items               *
*********************************************************
```

format lies in its simplicity. Care must be taken when adding features or it may lose some of its appeal. Unlike the typical menu-driven system, users may enter commands or interdependent command groups in random order without following a prescribed pathway.

```
C> RUN ACCT-SYS
Initializing Acct-Sys...
==> DIRECTORY
customers {cust#,name,street,city,phone}
items {item#,descrip,price,stock_level}
orders {cust#,date_ord,item#,qty}
acct_bal {cust#,curr_bal,amt_due,date_due,status}
payments {cust#,amt_paid,date_paid,pay_type}
==> LIST CUSTOMERS{CUST#,NAME} WHERE SORT={NAME}
c40023    bennett, carrie r.
c10009    cameron, michael w.
s78110    davidson, john q.
h98374    fuller, mark l.
f00485    garrett, david m.
k00917    larson, margaret c.
j00101    masterson, andrew h.
j00310    thompson, jane j.
j18375    williams, bill d.
==> ADD CUSTOMERS ITEM
cust #: J00803
name: MARTIN, JAMES M.
street: 503 QUAIL RUN LANE
city,state,zip: BOSTON, MA  20387
phone: (451) 665-7301
==> LIST ORDERS{ALL} WHERE {CUST#}=F00485
cust#       date_ord      item#     qty
c40023      3/25/88       100-1a    12
c40023      3/25/88       200-2a    5
c40023      3/25/88       k1011     2
c40023      5/10/88       3046-b    1
==> LIST ACCT_BAL{ALL} WHERE {CUST#}=K00917
cust#    curr_bal    amt_due    date_due    status
k00917   635.12      20.00      9/15/88     current
==> LIST ITEMS{ALL}
item#       descrip          price      stock_level
100-1a      crystal vase     39.95      32
200-2a      frame, 5x7       12.50      53
23684       fire screen      55.00      12
3046-b      elec skillet     29.95      44
h776-3      TV, 19" bw       89.95      21
k1011       luggage, 3pc     120.00     8
```

Figure 2. Command-Driven Interface

It should be noted that these descriptions reference only the fundamental design characteristics. These designs may be expanded to include additional features or visual display techniques. In many cases, the variety of data types and structures utilized within a single application warrant the formation of an interface which combines attributes from both basic interface types.

3.0 Selection Criteria

What specific factors should be evaluated to determine best fit for an interface? Which interface type best corresponds to each factor? A checklist of the major factors to be considered and a discussion of their relevance to either menu-driven or command-driven systems follows. A discussion of the classifications of user/DBMS interactions as presented by [Goldstein(2)] provides a similar framework in several areas. It should be kept in mind that such a discussion must be conducted in a generic sense. The wide range of manipulative techniques used in interactive interface design prohibits any rules from being designated as absolutes. In addition, factors should be evaluated in relation to each other and the overall best fit should be sought.

Comparison Factor	Menu-Driven Interface	Command-Driven Interface
1) Cost / Schedule	Relatively more expensive due to increased lines of code needed to handle I/O from interactive screens; screen formatter may have to be purchased	Less expensive for opposed reason
2) User Profile	Well suited for users who access system infrequently; screens often include text giving options or help	Best suited for user groups who use system often enough to familiarize themselves with system commands
3) User Training	Users can often follow menus and screens in a self-guided atmosphere	May require direct training and/or a well-prepared user manual to provide documentation of commands and syntax
4) Visual Image	Formatted screens are more visually expressive, menus allow system functions to be viewed clearly	May lack visual impact
5) Flexibility / Complexity of Data Access Requests	Screen I/O and access paths are set in design; complex data I/O functions can be set up in advance	User may devise any combination of valid commands to formulate ad-hoc data requests
6) Data Entry Requirements	Preferable if a variety of data types must be entered simultaneously; error detection done screen by screen	Data may be entered in streams; very efficient for singular data types; error detection usually performed on group basis
7) Structure of Data	Well suited to data that is logically oriented as single records	Allows display of large amounts of data at a time, preferable for data grouped in blocks/files
8) Security / Information Hiding	Screen formats allow only specific data values to be viewed	Access is less restrictive in general, data access requests are usually generic

Table 1. Checklist of Factors

In many instances, diverse user needs will be inconsistent with each other. The desire for easy access to the database for update purposes may conflict with data security requirements, for example. A difficult process of compromise, tradeoff, and refinement of requirements is needed to insure that the design or acquisition process has taken into account all significant user needs. [Stieger(3)]

3.1 Cost and Schedule

In very restrictive cases, this factor may tend to determine a great deal about the application in addition to the interface type. Limitations in cost and schedule realistically result in a scaling down of the proposed database functions and features. Consequently, the database I/O requirements are likely to be similarly reduced in scale and complexity. It

should be noted that the physical size of the database, in terms of the number of elements it contains or its physical storage requirements, is not a significant cost factor [Boehm(4)]. It is the complexity of the design, not physical size, that must be reduced.

The use of a command-driven interface would fit well into such an environment, since a key attribute of this interface type is its simplicity. Such an interface system can be operated with a relatively small amount of software, with a corresponding savings in effort and cost.

Many database management systems include a resident screen formatting package. Independent screen formatting packages may also be used. Such pre-packaged programs provide for quick and easy generation and manipulation of menu-driven screen images. Built-in features do most of the work needed for the basic tasks of defining a screen format, reading and writing data from the fields, and performing simple data validation. If the desired database design calls for simple, easily effected data access then such tools allow for a basic menu-driven system to be constructed quickly and efficiently. However, unless such a tool is already available in-shop, the cost of the screen formatting package must also be considered.

Menu-driven systems are also capable of handling the more complex I/O requirements allowed within a less restrictive environment. It follows that the code required to manipulate the interface must in turn be more advanced. A limited survey of several menu-driven systems revealed a rough order of magnitude estimate of the number of lines of code required to control a given number of interactive screens. An approximate average of 500 to 800 lines of code per screen was observed in the construction of a complete interface for the applications surveyed. It should be noted that the systems examined were fairly complex in terms of input and output formats, data validation requirements were extensive, and screen formats and features were moderately advanced.

It should not be assumed that the selection of a command-driven or simple menu-driven interface will sufficiently meet the cost restrictions on its own and thus eliminate the need for a trimming of any other database features. Such an assumption would be self-defeating. If the complexity of the database I/O is retained, the effectiveness or fit of either of these interfaces would be negated. To best meet cost and schedule restrictions and still generate an effective database tool both decisions must be made in tandem.

3.2 User Profile

An interface's effectiveness is dependent on its ability to satisfy the needs of the people who use it. The ease with which the user can communicate with the interface is an important consideration in the overall application design. How well this goal is met is often determined on a subjective basis by the users themselves. To assure a successful outcome, the designer or manager must be able to examine the application from the viewpoint of the customer.

User experience in data processing must be considered. If the majority of users are not data processing professionals, special emphasis should be placed on the need for clarity, straight- forwardness of operation, and especially on the detection and interception of errors. Conversely, if the users are knowledgeable about data processing, they will be better able to cope with greater system complexity with less explicit or elaborate error messages. [Stieger(5)]

An application which will be used in an intensive manner deserves an extra measure of care in the selection or design of the interface. Examples of this type of environment might include data entry applications for an accounting department or an order placement system for a mail-order firm. A primary concern of a typical user in this environment is efficient use of their time. Rapid completion of the most frequently performed tasks is essential. A command-driven system provides the users with the ability to rapidly access and execute any command. The familiarity gained from constant exposure to the interface allows users to operate the system from memory, without the need for on- line directions.

User populations which will associate with the database on a casual basis can be expected to require additional guidance in the operation of the database. Users who work with the interface on a limited scale will not have the benefit of repetition to keep their skills honed. [Goldstein(6)] A computerized card catalog system for a public library or an executive's monthly budget status inquiry system are typical examples. The average user may access the system only occasionally or may be completely unfamiliar with it. Systems of this nature must place more emphasis on tutorial aspects rather than speed. A menu-driven system's screen displays assist the user by providing him with a list of his available options and then guiding him through the task in a structured manner. The screen images clearly display the data values to be entered and may include helpful or descriptive text.

An experimental case study conducted in a real-world office environment revealed some interesting facts concerning user preferences relating to database interfaces. Although some trends were apparent, it was also found that there is a great deal of individual variation in terms of interface preferences and patterns of use. Although the menu-driven interface was the preferred interface for new users and became less frequently used the more experience a person had, its use never stopped completely. According to survey results of the users involved in the test, they turned to the menu-driven type of interface when they attempted to use new or unfamiliar parts of the system or when they needed to refresh their memories after an absence from the system. After experience was gained, the command-driven interface became the most frequently used. [Hiltz(7)]

Although it would not be practical to provide systems which include both types of interfaces, as was done for the experiment, the insights gained from the responses of users who were given the opportunity to use either as they chose can be very beneficial. The target user population should be evaluated in terms of their relative level of experience over the expected lifespan of the proposed system. If the database is expected to be used fairly consistently over a long period of time, users would be more likely to be satisfied with a command-driven interface - even though it may be more difficult to work with initially. While a menu-driven system might work very well for the first few weeks as users get acquainted with the system, they may soon outgrow it and spend the next several years wishing for a more concise interface. The opposite reasoning can be used to justify the selection of a menu-driven interface in instances where the database has a relatively short life expectancy or the users' exposure to the system will be limited or sporadic.

3.3 User Training Required

As indicated in the previous section, a comprehensive menu- driven interface allows the user to follow the menus and screens in a self-guided atmosphere. The user is not required to have any previous knowledge of valid commands or syntax. Options are generally indicated by the entry of a function key or numeric value whose purpose is identified on the screen. The use of menus has the cognitive effect of helping the user to develop a mental map of the structure of the system and the relationship among components [Hiltz(8)].

For many applications intended to serve the "general public", the use of a well-planned menu-driven system can enable a user with no computer literacy to accomplish the desired task the very first time he encounters the system. Millions of people each day do a significant part of their banking through computerized transactions on automated teller machines without having had any training whatsoever.

For more complex applications or those whose users can be expected to have had some exposure to the system, a brief initial explanation of the system and/or a concise supplemental users manual can be all that is required to make the users comfortable with the system. In these cases, the user is often reasonably familiar with the system and only needs help while performing the least familiar tasks or may need only a reminder or refresher when entering the system after a long absence. This allows the interface to become more streamlined by eliminating the need to instruct the user on each and every keystroke, while still providing a reasonable amount of guidance.

In a typical command-driven system the user may be provided with little more than a command prompt on the screen. To be able to provide the system with a usable directive, the user must have previous knowledge of the system commands. To acquire such information, users generally require direct hands-on training under the guidance of an instructor or, at a minimum, the availability of a comprehensive users manual. Although the initial investment of effort and time required to conduct the training and to generate the manual may seem to accomplish little in terms of actual production, it must be weighed against the long-term benefits of the increased efficiency of the trained user.

3.4 Visual Impact

Although it may be argued that the visual attributes of a system do not have a direct effect on its productivity, it cannot be denied that user perceptions and acceptance are affected. This is of particular importance in the development of a commercial system whose visual attractiveness may be directly related to its marketability. The same applies to an unsolicited system developed internally which may have to be demonstrated to a number of management or user groups for their approval. The primary goal of such a presentation is to demonstrate the greatest amount of system capability within a very short time frame.

A menu-driven system is perfectly suited to this particular situation. The use of the menus allows a user to clearly see before him a list of the functions provided by the system. The data screen formats instantly provide an overview of the data input and output values. A primitive demonstration prototype of the system can be constructed by simply formatting the proposed screens and linking them together. Another advantage gained by the use of fixed format, full-page screens is the ability to add artistic touches to the screen by the use of graphics, logos, borders and colors. An attractive, visually expressive interface can be invaluable in making a good first impression.

Visual impact is not a strong point for most command-driven interfaces. However, if the driving concern is functionality and the users are not influenced greatly by aesthetics or a "pretty" appearance then a command-driven system will not be adversely affected by this factor.

3.5 Data Access Requests

Applications may differ in the nature of the data access requests they must process. As in the case of a command-driven system interface, users may formulate ad-hoc data requests by combining any of a number of valid command words and formats. The interface is designed to recognize a list of defined key words and to interpret them when used in an established command language syntax. Although the command syntax is known when the interface is built, the actual commands themselves are not defined until the end user enters them. Command-driven interfaces thus allow for a great deal of flexibility in the user's manipulation of the database. Such flexibility would be a major advantage in a personal data management tool or the executive status query application mentioned earlier. Caution should be exercised, however, not to attempt implementation of an overly complex command syntax. The amount of code required to translate increasingly advanced commands rises on a progressively steep scale and user learning curves are extended.

Menu-driven interfaces are typically more structured in terms of the data manipulations. Such a structure can help a user who is not familiar with the internals of the database to be more comfortable in their use of the system. For those instances in which all required input and output values are known in advance a menu-driven system can be built to meet those specifications exactly. Complex operations can then be built into the screen hierarchy and performed by the user without undue effort. Code supporting the interface can be used to gather data values from several data files for display on the same screen or to display temporary, calculated values such as running column totals. By following the established access paths inherent in the menus a user who is not trained in database or information processing skills can still perform advanced data manipulation tasks.

3.6 Data Entry Requirements

In many instances, the primary function of an interface is to provide a means of loading data into the database. Occasionally large volumes of data may be added mechanically, such as a direct file load from disk or tape by a software package. More likely is the interactive entry of data by a person. The rate or volume of data entry expected, the variability of data types, and error detection requirements can each vary widely and should be reviewed in terms of the interface selected. As the volume of data which must be entered increases, so does the need to optimize the interface to best fit the characteristics of the data.

A command-driven format lends itself easily to the streaming of input values. That is, the input of a variable length, continuous list of values. If large numbers of homogeneous values are typically entered together, streaming can improve data throughput for entry intensive tasks.

Non-homogeneous data profiles are generally more adaptable to a menu-driven interface. Such profiles can include a diverse group of data types and formats (characters, text, integers, reals) that are logically grouped together or an entry format which is very restrictive (exact spacing or location of data fields). The data entry task can be clarified by the physical placement and dimensions of the fields on the menu-driven screen.

In keeping with the ability of command-driven interfaces to process large groups of data as a unit, error detection may also tend to be performed at less frequent intervals. Rather than interrupt the input of a stream of data the designer may prefer to do an integrity check after the entire data block has been entered and to notify the user of any errors found so that corrections can then be made. Given the physical groupings of data values that occur on a menu-driven screen, error detection is most efficient if handled on a screen by screen basis. It would not be practical to detect errors after a user has exited from a particular screen since it may be difficult to enforce his return to the same screen at a later time.

3.7 Structure of Data

As indicated in the previous section, data that is logically oriented in a block or group format is well suited for display by a command-driven system. Variable amounts of data may be displayed in a flexible, open-ended manner by the use of the scrolling features common in a command-driven environment. Large volumes of data may be reviewed rapidly. The user may freeze the scroll at chosen intervals to examine certain data more closely and then continue scrolling when desired.

Data which is comprised of a large number of smaller, independent records or units can most easily be maintained through a menu-driven interface. The screen displays allow the user to select and view a particular record or item quickly, without being distracted by the display of a series of non- selected items. If transactions such as deletions or modifications are performed at random or scattered intervals, the record which is currently being acted upon is always very clearly displayed.

3.8 Security / Information Hiding

The database administrator may wish to limit the access provided to the user community, particularly if it is a large and diverse group. As the size and complexity of the database itself grows, so does its utilization by various user groups. Both of these occurrences increase the need for data security. The database interface is the first line of defense against improper access.

Menu-driven interfaces by their very nature provide a simple access control. Data I/O is strictly governed by the designer through the format or fields that are included on each screen. Access paths are controlled by the menus provided. The structure of the interface prevents the end user from deviating away from the established access patterns.

Command-driven interfaces, on the other hand, allow a good deal more freedom in the way the end user accesses the data. By allowing the user a greater degree of control over the command sequences to be performed, data access is less restrictive. In addition, the data request results are typically more generic, allowing for the display of a greater volume of data. While this may tend to simplify the command language, it does not lend itself well to information hiding.

4.0 Reconciling Factors

As discussed, several different characteristics can have an impact on the fit of a database interface. While they may seem to be very different from each other, in most cases the factors are interrelated and tend to support one another. For example, the data security and information hiding concerns described relate quite closely to the corresponding user profiles. Just as command-driven users are typically more knowledgeable about the structure of the database they are working with, often using the database as a personal management tool, so the need for tightened security is diminished. A menu-driven system's capacity for more secure operation is useful when it is made available to a broad group of users who may not be too familiar with the system. Other supporting relationships can be found between the data entry characteristics and the physical structure of the data.

In evaluating or planning the proposed database environment, the database manager or designer will frequently discover that while some aspects of the environment may tend to indicate the use of one interface type, other aspects will indicate use of the opposing interface type. To resolve these apparent contradictions the designer must employ creativity to develop a hybrid interface which meshes the desired characteristics of both interface types. In practical, real-world applications there is often some degree of variance in the alignment of the factors, particularly for those applications whose functional scope is relatively broad. In other words, most environments do not fit perfectly into either a menu-driven or command-driven category. While the use of a commercial off-the-shelf package will limit the amount of customization that can be achieved, a database which is built in-house or under contract should be able to meet almost any combination of needs.

In order to aid in the reconciliation of the various factors presented and to provide a quantitative tool to assist in the selection process a worksheet has been provided. The worksheet is intended to simplify the task of categorizing the particular aspects of the designers' environment and to arrive at a composite characterization. A brief analysis and recommendation are provided based on the numerical results of the survey. The results can provide a valuable tool in the decision making process.

Interface Environment Survey

Below is a worksheet designed to help you identify the application environment within which the interface you choose must perform. To use the worksheet, you will assign a numeric value to each of a series of project descriptions based on the degree to which they accurately reflect your environment.

Each factor is assigned a value and two differing project descriptions are given. If your project very closely matches one of those descriptions, the total value for that factor should be recorded beside the description. In some cases, however, certain aspects of both descriptions may apply. You should then distribute the total value across both of the descriptions proportional to the degree that each reflects your particular situation. The relative weighting values assigned to each factor were subjectively derived and may be subject to refinement as further field data becomes available. The total value assigned to each pair of descriptions should always equal the weighted value indicated beneath the factor name.

The survey has been applied as a postmortem exercise against a limited number of database systems, including those systems which provided the basis for the data used to initially calibrate the survey. These preliminary validation tests have yielded favorable results, producing a reasonable and relatively stable correspondence between the attributes of the database systems as they were implemented and the survey's forecast. Currently, it is too early to have collected any data regarding the performance or fit of systems for which the survey was used as an input to the design process. Such data will be evaluated and incorporated as it becomes available.

	A	Pts.	B	Pts.
Cost / Schedule 20	- Scope of project is small or has been limited by budget / time restrictions - No screen format package is available in-house	___	- Scope of project is relatively advanced and budget is sufficient - Project scale is small and a screen formatter is already available	___
User Profile 30	- Users will access system on an infrequent or casual basis	___	- Users work with system on a daily basis	___
User Training Requirements 10	- User community can be identified and gathered together for formal training sessions - User manuals can be distributed to users or to all workstations	___	- Formal training sessions are impractical - Users may not have any computer background at all, lack of computer literacy	___
Visual Image 10	- System functionality is more important than its appearance	___	- System is unsolicited and so its marketability is a concern - During demos, system functions must be easily illustrated - Rapid system prototype is needed as design tool and to get project go-ahead	___
Data Access Requests 20	- Exact I/O is not known at time of system design - User needs to control I/O as in a personal data management tool - Ability to generate ad-hoc data requests quickly is very desirable	___	- All I/O requirements are known in detail at design time - Very complex data manipulations are required and must be closely controlled to ensure accuracy	___
Data Entry Requirements 20	- Large volumes of data of a single type are common - Error detection can be performed on large blocks of data at once	___	- Data contains a mix of several data types or formats - Spacing or location of data values is important - Data must be monitored closely for errors and they must be corrected quickly	___
Data Structure 30	- Data is logically organized into blocks or files such as text	___	- Data can be divided into small independent units or records	___
Security / Information Hiding 10	- User community is small and is knowledgeable about the system; chances of corruption are reduced - Users require unrestricted access to data to do their job most efficiently	___	- System needs to be protected from possible corruption or error introduction by unskilled users - Database is used by a wide range of users whose access may need to be selectively limited to certain sections of the database	___
		Totals ___		___

Analysis Of Results

The column totals computed in the survey will be used to determine the best fit of an interface type to the environment described. The totals provide a standardized, quantitative way to identify the selection factors present. A total of 150 points is available for distribution between the two columns. Of course, a total of 150 in either column would indicate a perfect fit and would make the selection process automatic. However, this can seldom be expected to occur. More likely are cases which include some aspects of both the interface descriptions. This means that in most situations some compromises must be made.

To standardize the selection process and make it more reflective of the real world, a certain delta value away from a "perfect" fit must still be accepted. Therefore, a 70% proportion of fit for either interface will be considered as a suitable environment for the successful implementation of the indicated interface type. This implies that if one of the column totals generated above represents at least 70% (105 points) of the 150 points available then that interface type will be selected. If neither column totals 70% of the available points, then a special situation exists which is best resolved by a combination of both interface types.

Column A > 105

Based on the environment description given for the application, the best fit will be achieved using a command-driven interface.

Column B > 105

Based on the environment description given for the application, the best fit will be achieved using a menu-driven interface.

In both of these cases, those factors which were applicable in the column with the lower total should be kept in mind during the design of the system and its interface. In many cases, the concerns described which seem to indicate the use of a different approach can be handled by making some minor design concessions, modifying the project requirements or, in some instances, by modifying the environment itself. On the whole, though, the basic interface type indicated should meet the overall needs of the project quite well.

Column A & B < 105

In this case, the best interface fit for the application described is a combination of the features found in both command-driven and menu-driven systems. Although such design tasks may require an extra measure of creativity and skill, the result will provide an effective and productive database management tool.

Some typical examples of combined interface features include:

- The addition of extra user help messages or instructions as a part of the normal command-driven display.
- Modification of some menu-driven screens to allow them to accept entries or display data in a streaming or scrolling fashion.
- Construction of the command-driven system to include a greater degree of control over the commands allowed at a certain time. Commands which are valid may also be dependent on the identity of the user.
- Greater flexibility may be built into a menu-driven system by providing a mechanism, such as a keyword or command field on the screen, which will allow a user to call up the next desired screen directly rather than progressing through the menu hierarchy.
- For a menu-driven system which must be able to process certain complex data transactions, those transactions can be coded to the greatest degree possible as a part of the interface software. The coded or automated process can then be made available to the user through the use of a specialized command similar to the other user formulated commands.

These are only a few of the many ways in which the base type of the interface can be expanded to meet the needs of almost any project.

5.0 Summary

When designing an interactive database application, the user interface deserves particular attention. The interface is the link between the information contained in the database and the people who must use that information. The physical database itself may be no more than an abstract, nebulous cloud of data to the users. The interface provides a way for them to associate a meaningful, visible structure with that data. The primary purpose of the interface is to provide a way for the user to efficiently access a selective subset of data while protecting the integrity of the entire database.

In general, two fundamental types of interfaces are common. A command-driven interface provides a flexible interface which can provide efficient data access for a well-trained user. A menu- driven interface offers a structured, easily understandable way for even casual users to perform a wide range of database operations. Many applications are best served by a hybrid interface which incorporates aspects of both basic interface types.

To determine the interface type which will best fit the needs of a given application environment, several factors must be considered. The major areas to be examined include the primary purpose the database will serve, characteristics of the typical user, and the physical and logical structure of the data. These can be specifically defined as follows: cost and schedule restrictions, user population profile, user training requirements, visual impact of the interface image, complexity of data access requests, volume and variability of data entry requirements, physical structure of the database files, and the robustness of the security measures required.

This article and survey are intended to serve as a baseline for further exploration of the unique circumstances and needs found in your particular office or application. In addition, input from other sources and further research should be included in order to reach a concensus decision which can be acted upon with confidence.

BIBLIOGRAPHY

1. J.C. Thomas, "Organizing for Human Factors", **Human Factors and Interactive Computer Systems,** Proceedings of the NYU Symposium on Computer Interfaces, May 1984, New York, NY, pp. 29-45.
2. R.C. Goldstein, **Database Technology and Management**, John Wiley & Sons, Inc., New York, NY, 1985, pp. 217-234.
3. "Selection and Acquisition of Data Base Management Systems", Codasyl Systems Committee, W.H. Stieger, Chairman, March, 1976, p. 34.
4. B.W. Boehm, **Software Engineering Economics,** Prentice Hall, New Jersey, 1981, pp. 386-389, 642.
5. "Selection and Acquisition of Data Base Management Systems", Codasyl Systems Committee, W.H. Stieger, Chairman, March, 1976, p. 37.
6. R.C. Goldstein, **Database Technology and Management**, John Wiley & Sons, Inc., New York, NY, 1985, pp. 217-234.
7. S.R. Hiltz, **Online Communities: A Case Study of the Office of the Future**, Ablex Publishing Corp., New Jersey, 1984, pp. 121-122.
8. Ibid.