Design rationale of the AS/400 user interface

by J. H. Botterill

This paper discusses the design rationale of the software user interface of the Application System/400™ (AS/400™). It presents the design approaches used to produce the interface of this interactive system.

Although advancements in technology are making more and more complex systems available to the users of small systems, the interface between the user and such systems must become simplified in order to fully utilize the richer function provided. The small business environment, in particular, requires that new systems be usable by the personnel in such an environment. The introduction of such machines cannot require the addition of new and sophisticated data processing expertise.

In the past, much of the perceived ease of use of smaller systems was due to their limited function and to the fact that their users were primarily professional programmers, operators, and data-entry clerks. These users were able to learn the system interfaces because they were trained as data processing personnel, and the interfaces involved relatively few functions. Today, end users want to directly access and manage their own data. Functional requirements for end users now include a database system, communications, security, backup, and problem determination. Even in large installations, users of the system usually own the data that they are utilizing. Therefore, limited function can no longer be a basis for ease of use. New design approaches and interface design standards are necessary.

The Application System/400™ (AS/400™) is a new system that spans the range of small-to-intermediate systems and is designed to meet the requirement of making system-provided function available to end users. It addresses the needs of a simple environment with a single user, as well as complex environments with many workstations and many users. Supporting nonprogrammable terminals[1] is crucial, due to their cost advantage and the customer's current investment in them. At the same time, the percentage of programmable workstations attached to systems will continue to grow. The mixed environment needs to be supported in a consistent manner to allow a graceful transition from one to another. Applications need to be able to fully capitalize on the programmable workstation capabilities where appropriate.

For these reasons, the AS/400 system interface is primarily a nonprogrammable terminal interface which is also available on programmable workstations. It is designed to be a simple, self-guiding interface for new users; an efficient, productive interface for experienced users; and the consistent base for addressing new user interface technology. The interface is similar to that available on stand-alone programmable workstations and is the result of the use of new advancements in object-oriented interfaces, object-action interaction, selection, layering, and word

entry on a base of proven ease-of-use techniques. It includes: system-wide consistency, user-friendly menus, self-directing entry displays, extensive help, powerful list displays, a comprehensive command set, and the consistency of an underlying object structure.

The primary focus of this paper is to discuss the design approaches used and the rationale behind the design of an interface which seeks to meet the objective of addressing a broad spectrum of users. A series of approaches are discussed, some of which add new dimensions to proven approaches, and others which are new. For example, system consistency includes not only consistency within the parts of a system, but consistency with other IBM systems, and consistency between nonprogrammable terminals and programmable workstations. Productivity and ease of learning are improved through object-oriented interface design and object-action interaction. The interface flexibility is addressed through layering, selection techniques, and fast paths. The use of word entry and search technologies improves the retrieval of help information. The paper addresses the user interface design approaches that were used in developing the system; it does not discuss the function, even though a significant amount of function is provided to make the users more productive. The paper concentrates on general, system-wide approaches, rather than how specific functions are designed. For details on the function or the interface for a particular function, the reader may refer to the AS/400 publications listed as general references at the end of this paper.

Thus, some of the approaches followed in designing the AS/400 user interface include:

- Consistency of appearance and operation
- Object-oriented design and object-action interaction
- Recognition and selection
- Interface layering
- Grouping
- Fast paths

The approaches themselves are described, along with how they have been applied to the two primary user interfaces of the system, interactive panels and commands.

The *interactive display interface* is menu driven and provides access to all interactive system function. This interface allows the user to access one or more objects of a given type and then perform actions on them, or to perform actions on a task basis. The panel dialog for this interactive interface is made up of full-screen panels and is described in the section, "Consistency of appearance and operation."

The *command interface* is a high-level action-on-object command interface called the control language (CL). It is a fast path for requesting functions from command lines on interactive panels, as well as a high-level programming language for controlling application programs and invoking system function. The design of the CL is described later in this paper.

The AS/400 provides a new dimension in user interface consistency.

Design approaches

Consistency of appearance and operation. The key to designing an effective user interface is to establish a set of rules that cause the application to respond in a consistent and easy-to-understand manner.

Just as in a dialog or interaction between two people, the dialog between a person and a computer application is much more efficient and comfortable if the responses fit what is expected and can be easily understood. The sooner a user can recognize the interaction style of an application and can know what type of responses to expect, the sooner that user can be effective. The user's view of the application is referred to as the user's conceptual model. The consistency of a new application and the similarity between it and other applications with which the user has experience, greatly reduces the time required to build the correct user conceptual model.

For these reasons, IBM has developed a consistent set of rules for the user interface of its AS/400 products. Users who have multiple AS/400 applications or use both applications and portions of the AS/400 operating system (Operating System/400™, or OS/400™), benefit greatly from user interface consistency. Less

learning time is required, and less time is required to readjust when switching from another application. Time and money are saved, user frustration is reduced, and job satisfaction is increased.

The AS/400 provides a new dimension in user interface consistency. It is a comprehensive, object-oriented interface that uses four primary panel types. The design of these panel types is based on the design approaches presented in this paper and the rules for a nonprogrammable terminal interface as specified in IBM's Systems Application Architecture (SAA) user interface document, *Common User Access: Panel Design and User Interaction*.[2,3] Since the AS/400 interface is based on the Entry Model[4] of the Common User Access (CUA) interface, it gives the added benefit of a level of consistency with new SAA products on attached Personal System/2® (PS/2®) programmable workstations and on System/370 computer systems.

The four basic CUA panel types used are menu, list, entry, and information. These as well as other application-specific panel definitions are built from a set of panel elements, panel areas, dialog actions, and function key assignments, many of which are CUA-defined. For example, F3 is used to exit, F4 to obtain prompting, and F12 to cancel back to the previous panel. The primary CUA elements that are not utilized in the AS/400 interface are action bars, pull-downs, pop-up windows, cursor selection, and mnemonic selection, all of which are normally used in PS/2-based applications and which take advantage of a programmable workstation and provide a windowed direct manipulation interface.

In this section, the design consistency conventions and rules are presented in terms of panel layout, panel areas, panel types, and common dialog actions and function key assignments.

Later sections of this paper address the higher-level design approaches used in the AS/400 interface that, together with the consistency rules, yield the design and style of the AS/400 user interface.

Panel layout. Four standardized panel types are used to establish consistency on the AS/400. These are:

- *Menu panels* that display a fixed list of choices from which the user can select an object, an action, or another menu
- *List panels* that display a variable list of objects and allow users to request actions to be performed on them
- *Entry panels* that contain fields in which the user can type information to qualify a request or enter data
- *Information panels* that display information for viewing only; these can be data presentation panels or help panels

All panels are one of these four basic types, except for a relatively small number of application-specific panels (such as calendar, editor, or screen design panels). Each of these four panel types follows a consistent basic layout which is designed to be scanned from the top down and from left to right. To confirm that the correct panel was selected, identifying information is positioned at the top. This is followed by instructions and the panel body, which is the primary interactive or information presentation area. Secondary elements are at the bottom, including a command area for fast path commands, function key descriptions for navigation and alternative functions, and a message area. This layout is shown in Figure 1.

Panel areas. The consistency across the AS/400 user interface is further enhanced by maintaining consistency within each area of the panel.

The *title line* contains two identifiers:

- The panel ID is a short identifier for the panel which is primarily used for menus. It is located left-justified on the title line. In the case of menus, it can be used as the target of the GO command to move directly to that menu.
- The text title clearly communicates the purpose and context of the panel. It is centered on the title line.

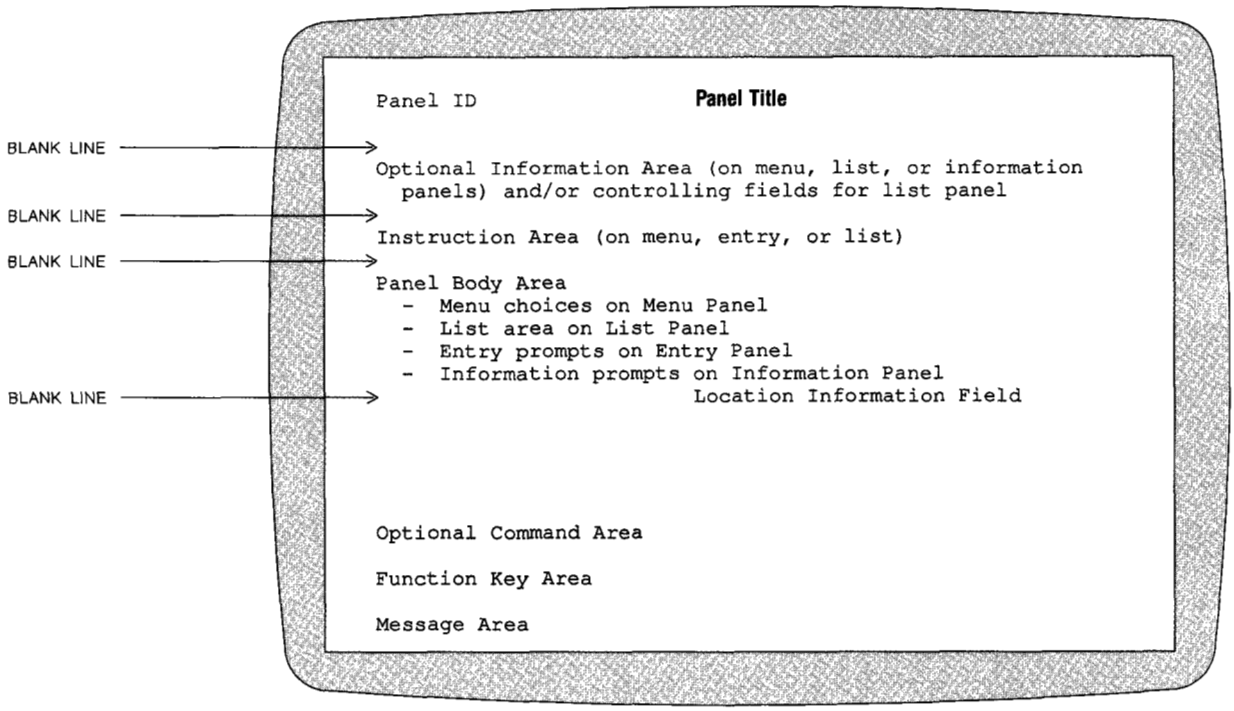
The *information area* contains identifying information relating to the content of the panel body area. It is shown below the title line in a separate area with labeled output values. For example, the information for a panel of the content of a job queue identifies the name of the job queue and other appropriate information, such as status. The top appears as follows:

```
                    Work with Job Queue

Queue: USERQ01    Library: QGPL    Status: RLS
```

The *instruction area* contains instructions that tell users what is required to interact with the panel body area. Instructions are kept short and address only

Figure 1 Common AS/400 panel layout



the primary interaction, so users read them and are not confused by detail. The function key area and help panels address the other secondary interactions that are supported.

The AS/400 uses very generic, consistent instruction text for each panel type. For example, on a menu panel the instruction is "Select one of the following"; on a list panel, "Type options, press Enter"; on an entry panel, "Type choices, press Enter"; and on an information panel, "Press Enter to continue." The text follows regular sentence style for readability, begins in position two of the line, and is preceded and followed by a blank line so as to be clearly visible to new users.

The *panel body area* is where the main user interaction and information presentation occurs. The other areas support it. For example, in a menu panel this area contains the choices, in a list panel it contains the list entries, and in an entry panel it contains the entry fields with their prompts. If all or a part of the panel body area contains more information than can be visible at one time, that area is made scrollable. The Page Down and Page Up keys are supported for scrolling forward and backward, respectively. Whenever the user is on any part other than the last part of a scrollable area, the word "More ..." appears in high intensity in the location information field, right-justified on the bottom separator line (see Figure 2). When the user is on the last panel of a scrollable area, "More ..." is replaced with "Bottom."

The *command area* contains an entry field where application or system commands can be typed as a fast path. The format of the command area is a prompt containing the word "Command" (e.g., "Selection or command"), followed by an arrow on the subsequent line and the entry field.

The *function key area* for the AS/400 describes the active function keys. The function key descriptions begin on line 22, if two lines are needed, and on line 23 if only one is used. The format of each function key description is: Fn=function, where "n" is the function key number. For example, F3=Exit.

"F24=More keys" is shown and active if all active function keys are not displayed. This may be for lack of space or because some assignments are for unusual situations which would be confusing to a new user. The F24 key results in the next set of function keys being shown.

The *message area* is where one or more messages are displayed to identify conditions relating to the entered values or requests.

These panel areas are then used in each panel type as appropriate.

Panel types. Four primary panel types are used across the system to develop a highly consistent predictable interface.

One of the four basic panel types is a *menu panel.* An AS/400 menu displays a list of choices from which the user can make one selection. A menu always has a title, an instruction, a list of choices, and a labeled entry field for typing the number of the choice selected.

A comprehensive set of AS/400 menus allows users to quickly identify and select the type of object to work with or the task to perform. The type of object may be a file, a document, a job, or mail, as shown in Figure 3. The task, for example, may be a general user task, an office task, an operations task, or a programming task. Some choices result in a lower-level menu, with a more refined grouping of choices. In this way, the interface can accommodate either object or task (action) requests. Object requests usually result in displaying a list of the requested type of objects to which the user is authorized. On the list panel, one or more actions can be requested on the displayed objects. Usually task choices go to a task-specific entry panel. In either the object or task case, the user need not know any commands, keywords, or option names. Where needed, the system presents an entry panel with multiple fill-in-the-blank prompts.

Specific menus are provided for common groups of tasks, such as office, programming, and operation tasks. A User Tasks menu provides access to common tasks (see Figure 4) for users who are not data processing professionals and do not need the full function of the other specialized menus. For example, such users may use an application and also need to send a message to a coworker (option three) or check their printed output (option five) without having to be trained as a system operator.

A command line is provided on each system menu. Individuals who use the system frequently can navigate directly to any menu by typing GO and the

Figure 2 Scrollable area with location information

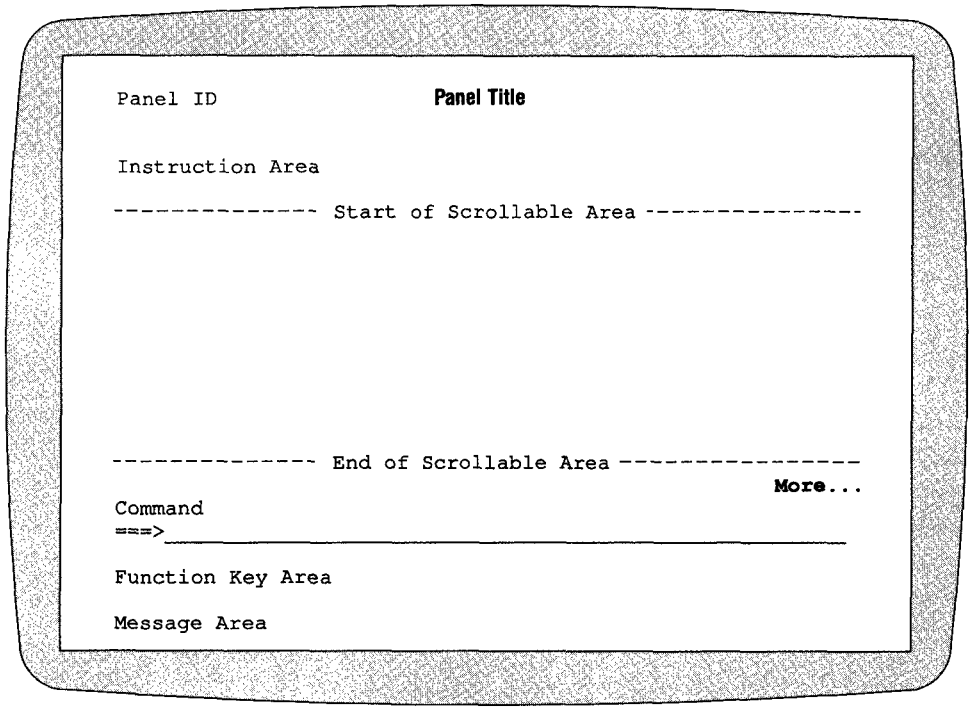


Figure 3 Interactive display interface

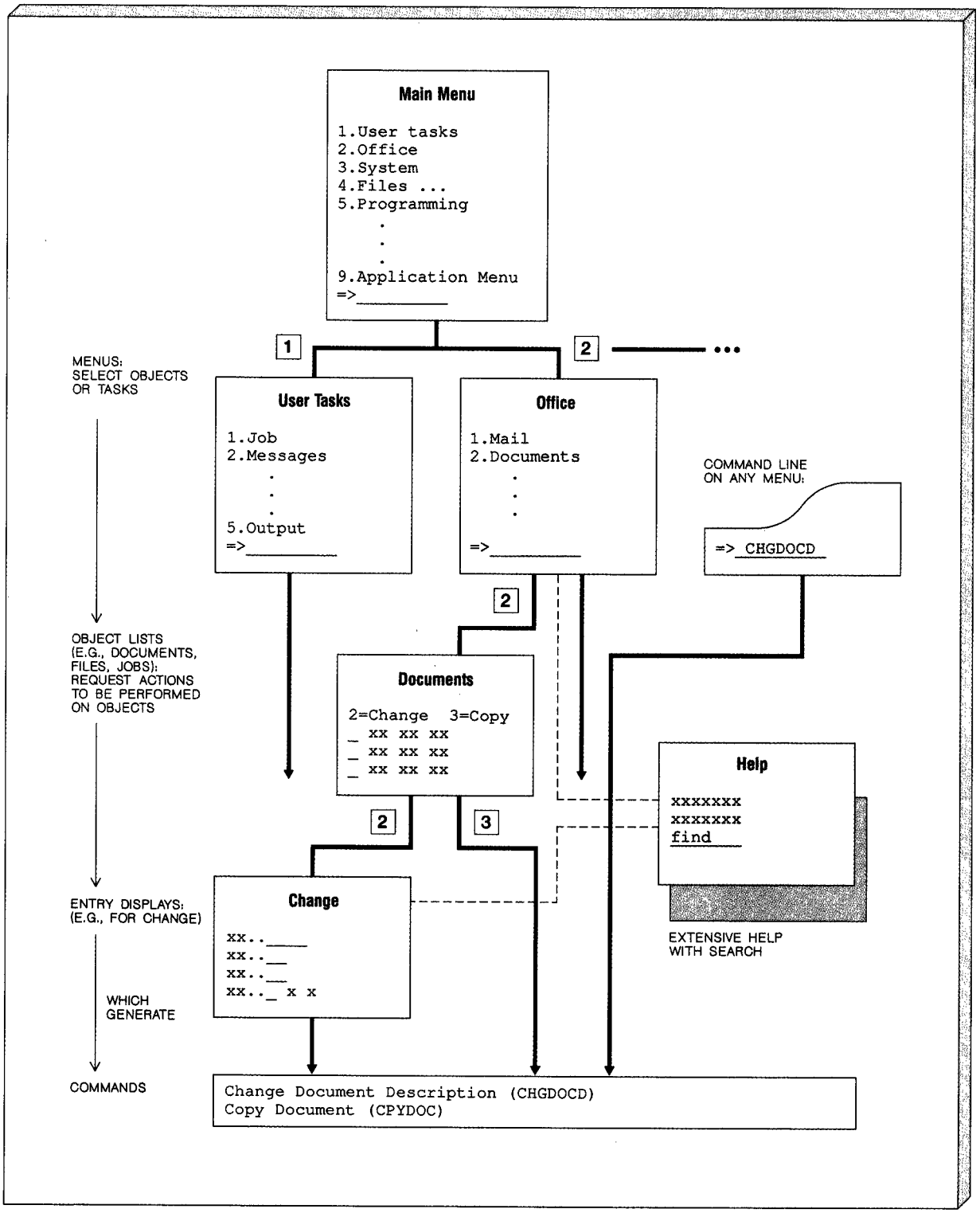
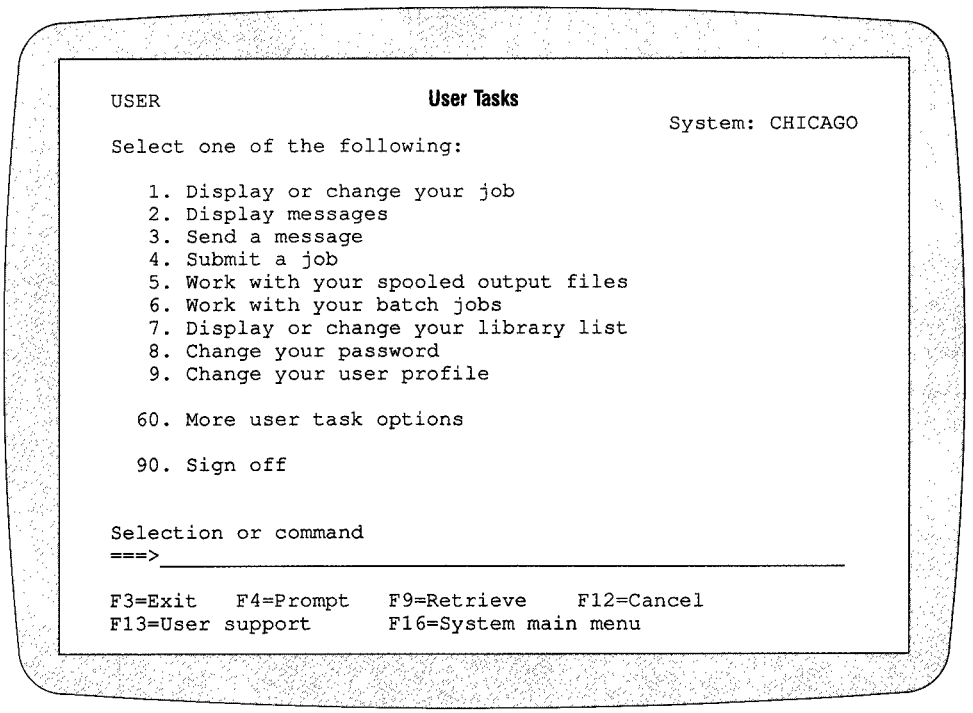


Figure 4 Example of a menu panel



panel ID that appears left-justified on the title line of the desired menu. The menu panel ID is a simple descriptive name, like MAIN, USER, or OFFICE. Other commands can be entered on the command line to request functions without using the menu option paths or leaving the current panel. For example, CHGDOCD entered on the command line of any menu (as shown in Figure 3) runs the Change Document Description function.

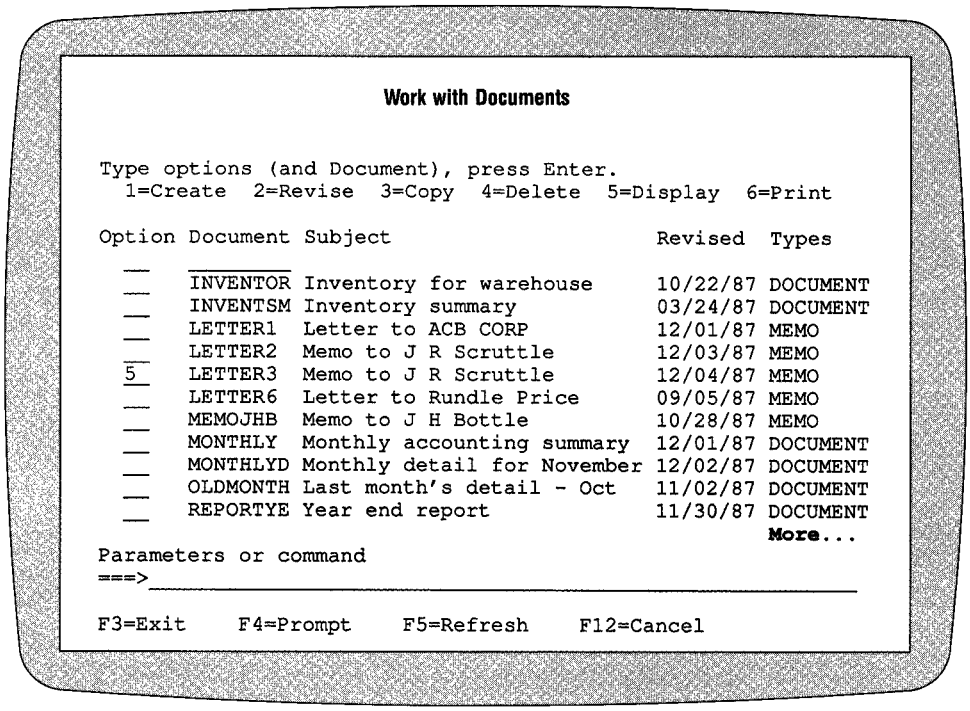
When a type of object is requested on a menu or by entering a command, a *list panel* is provided showing a list of the objects with type and attribute information. A list panel provides a convenient means to perform actions directly on objects, without having to recall and enter an object's name for each action. Actions are requested by entering an action in the option entry field in front of each object name. Figure 5 shows a list of documents with a "5" typed next to "LETTER3" to request a panel of the content of LETTER3. The supported action options are clearly visible in the upper instruction area. These list panels are referred to as "Work with" panels in menu options, commands, and list panel titles, because users can remain on them and focus their work on the set of objects presented.

In the key areas of data definition, query, and office, AS/400 introduces enhanced list panels with an input-capable list entry at the top of a list. This is a line under the column headings with an underscored entry field in each object-identifying column. In Figure 5 the list entry is made up of two entry fields, the option entry field and a document name entry field. Users can type the action option desired and the name of the document, without having to find the document in the list. The panel also allows a request to be typed to create a document that is not in the list, without having to leave the list area.

In cases where the list actions require further qualification and therefore present an entry panel, a fast path is provided, allowing specification of the parameters on the command line when entering action option numbers in the list area. The entry panels are bypassed if the required parameters are specified on the command line. The availability of this fast path is indicated by the prompt "Parameters or command" for the command line.

Entry panels, which allow users to fill in the blanks, are provided when more details are needed after a task is selected from a menu or via a command or

Figure 5 Example of a "Work with" list panel



```
                    Work with Documents

Type options (and Document), press Enter.
  1=Create  2=Revise  3=Copy  4=Delete  5=Display  6=Print

Option Document Subject                          Revised  Types
  __   ________
  __   INVENTOR Inventory for warehouse          10/22/87 DOCUMENT
  __   INVENTSM Inventory summary                03/24/87 DOCUMENT
  __   LETTER1  Letter to ACB CORP               12/01/87 MEMO
  __   LETTER2  Memo to J R Scruttle             12/03/87 MEMO
  5_   LETTER3  Memo to J R Scruttle             12/04/87 MEMO
  __   LETTER6  Letter to Rundle Price           09/05/87 MEMO
  __   MEMOJHB  Memo to J H Bottle               10/28/87 MEMO
  __   MONTHLY  Monthly accounting summary       12/01/87 DOCUMENT
  __   MONTHLYD Monthly detail for November      12/02/87 DOCUMENT
  __   OLDMONTH Last month's detail - Oct        11/02/87 DOCUMENT
  __   REPORTYE Year end report                  11/30/87 DOCUMENT
                                                          More...
Parameters or command
===>_____________________________________________________________

F3=Exit    F4=Prompt    F5=Refresh    F12=Cancel
```

by selecting an action from a list panel. Figure 6 shows an entry panel for a print request. The entry panels are straightforward and require minimal user interaction. They have a single column of entry fields, each preceded by a field prompt and followed by a list or description of the acceptable values for that field. The values can be numeric values for fixed, noncommand choices or actual command values, like *NO, for entry panels resulting from pressing F4 (Prompt) for a command (see Figure 11, later).

The panel body of an entry panel is made up of three columns from left to right. The columns are the field prompts, the entry fields, and the descriptive text that describes the values that can be entered in the entry fields. Each column is left-aligned, as shown below.

```
Type choices, press Enter.

  Prompt . . .  _____________  Values for field 1

  Long prompt   _________      Values for field 2
```

The user is asked to respond only to required and frequently used prompts. Choices that are infrequently used are not initially presented. They are presented on a following panel if it is determined, based on the initial responses, that more choices are indeed necessary. For example, if a Copy File request references a diskette file, only diskette-related options follow. Tape or database options are not shown. This is called "intelligent prompting." The prompts are tailored to user responses.

The prompt function is also layered. Less frequently used parameters may be requested by pressing the F10 (Additional parameters) function key. Each of these techniques results in users not having to analyze the individual fields or choices that do not apply to their task. Wherever possible, default values are already entered in the fields. The combinations of defaults are carefully chosen so they result in a commonly needed correct request.

The fields on an entry panel take two forms. The first form is an *entry field*, which requests a name or user-supplied value, like a document name, as shown in Figure 6. An underscore shows the field's maximum length. For certain entry fields that accept a name, the system makes available a list of the objects to which the user is authorized. "F4 for list" is shown

Figure 6 Example of an entry panel

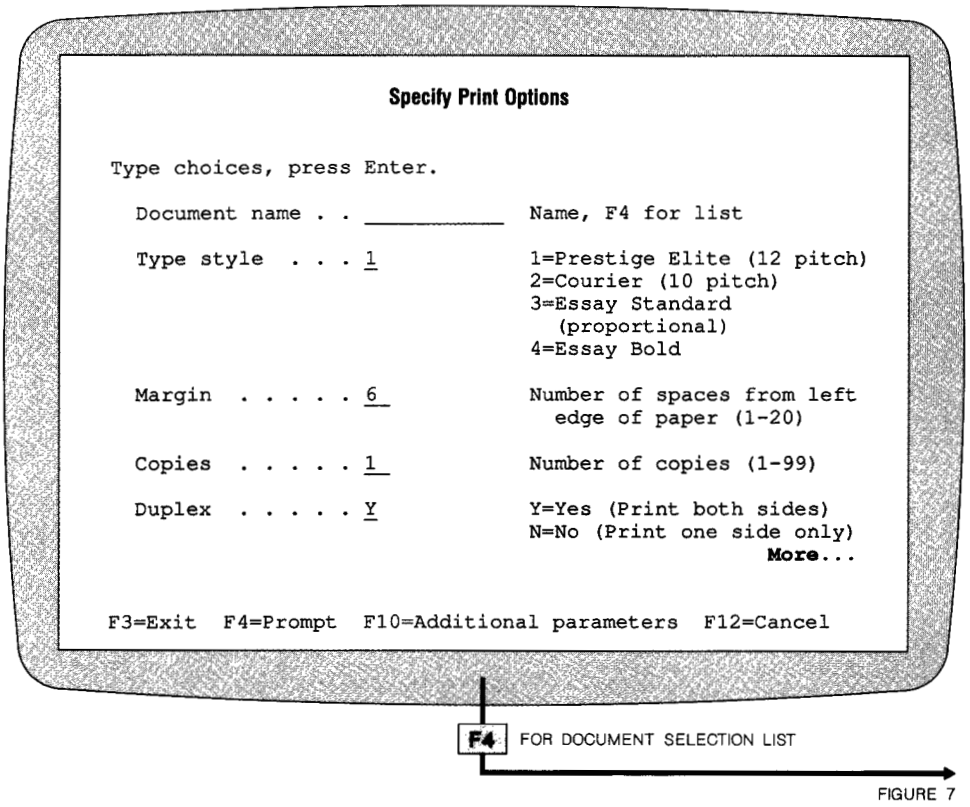


FIGURE 7

to the right of these fields to indicate that the F4 key will request the list, shown in Figure 7. The user can then make a selection from the list rather than type in the name.

The second form of field on an entry panel is a *selection field* that allows a selection from a fixed set of choices. The values are numbered as shown by the prompt for type style in Figure 6, unless the value itself has significance, as in the case of an entry panel for a command. The user need only type the number for the desired choice in the same fashion as on a menu. When the prompt requires a Yes or No response, Y and N are accepted for Yes and No, as shown by the duplex prompt in Figure 6. In the case of an entry panel for a command, the actual parameter values are accepted and are listed to the right of the entry field, like *REPLACE, *ADD, *MERGE.

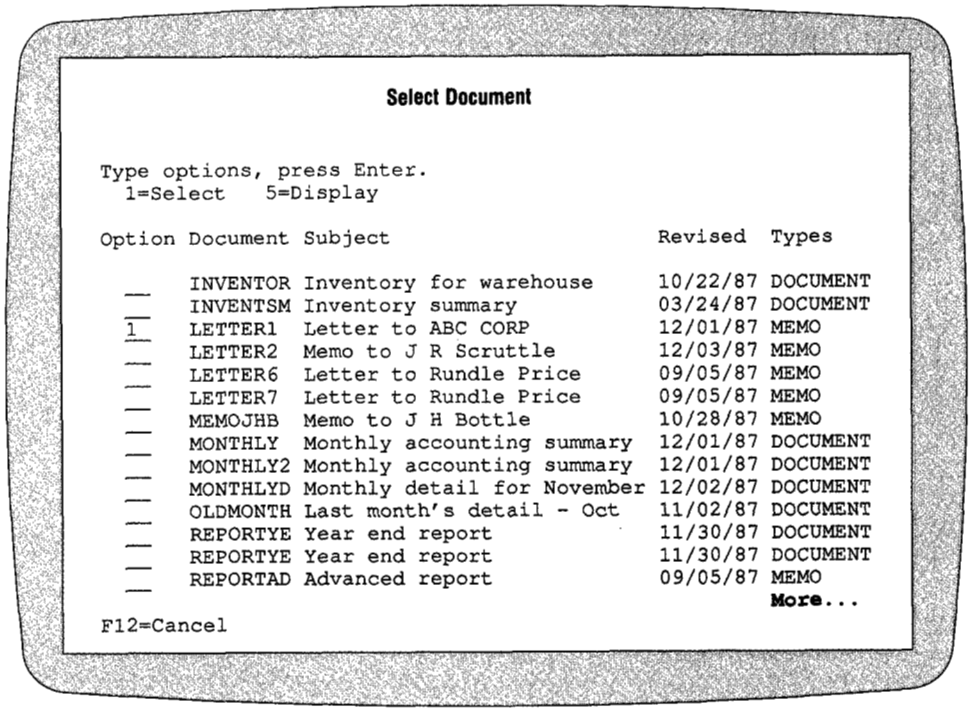
Information panels display protected information. The information panel shown as an example in Figure 8 displays a series of output fields that are identified by field prompts. The format is very similar to an entry panel because the user is probably already familiar with that format from entering or changing the information. This avoids having to learn and associate two formats.

Field prompts are in regular sentence style and are located to the left of the field they identify. Field prompts for output fields are not preceded by an instruction line. The colon after the prompts and the lack of an underscore indicate that the values are output only.

Object-oriented design. In addition to consistency of appearance and operation, one of the primary design features of the AS/400 is its object-oriented approach. Objects are the means by which information is stored and processed. They are named collections of data and attributes that are visible at the user interface. The internal representation of the data and attributes is not visible. The functions of the system operate on the external objects.

The external objects on the AS/400 include conventional data collections such as files and programs, as

Figure 7 Selection list



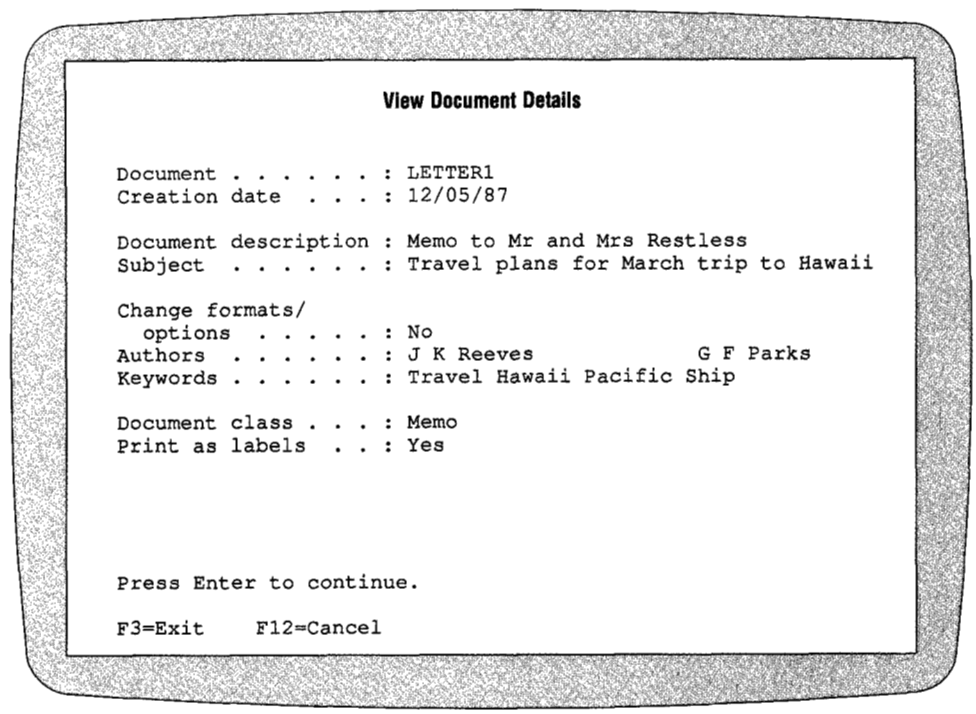
```
                    Select Document

Type options, press Enter.
  1=Select   5=Display

Option Document Subject                          Revised  Types

  __   INVENTOR Inventory for warehouse          10/22/87 DOCUMENT
  __   INVENTSM Inventory summary                03/24/87 DOCUMENT
  1    LETTER1  Letter to ABC CORP               12/01/87 MEMO
  __   LETTER2  Memo to J R Scruttle             12/03/87 MEMO
  __   LETTER6  Letter to Rundle Price           09/05/87 MEMO
  __   LETTER7  Letter to Rundle Price           09/05/87 MEMO
  __   MEMOJHB  Memo to J H Bottle               10/28/87 MEMO
  __   MONTHLY  Monthly accounting summary       12/01/87 DOCUMENT
  __   MONTHLY2 Monthly accounting summary       12/01/87 DOCUMENT
  __   MONTHLYD Monthly detail for November      12/02/87 DOCUMENT
  __   OLDMONTH Last month's detail - Oct        11/02/87 DOCUMENT
  __   REPORTYE Year end report                  11/30/87 DOCUMENT
  __   REPORTYE Year end report                  11/30/87 DOCUMENT
  __   REPORTAD Advanced report                  09/05/87 MEMO
                                                          More...
F12=Cancel
```

Figure 8 Example of an information panel



```
                  View Document Details

Document . . . . . : LETTER1
Creation date  . . . : 12/05/87

Document description : Memo to Mr and Mrs Restless
Subject  . . . . . : Travel plans for March trip to Hawaii

Change formats/
  options  . . . . . : No
Authors  . . . . . : J K Reeves              G F Parks
Keywords . . . . . : Travel Hawaii Pacific Ship

Document class . . . : Memo
Print as labels  . . : Yes




Press Enter to continue.

F3=Exit    F12=Cancel
```

well as those unique to the AS/400 such as job descriptions and message queues. All data are stored on the system in object form and are processed through interactive panels, control language commands, and the high-level programming languages.

Objects are brought into existence through a Create command function that defines the name, attributes, and initial value or values for the object being created. Each object is assigned a type which is determined by the object's specific purpose. After an object is created, it remains on the system until it is explicitly deleted by a Delete function. During its existence, only operations that are valid for that type of object are allowed to be performed on the object. Only users that are authorized to the specific object and to the specific operations can perform those operations on the object.

The key advantage of the object-oriented design is that users only see and specify attributes which are meaningful externally. The internal structure and actual storage occupied by the information are hidden. Users do not have to know if a given object is implemented as multiple data structures or one. They do not have to know offsets or internal representation. For example, a database file is made up of four machine object structures: a space, a cursor, a data space, and a data space index. The system manages the individual pieces of the file in a way that allows users to perceive the file as a single object.

To minimize user learning, all objects have a set of common attributes including: name, type, subtype, library, creation date, last change date, and text description. They also have a common set of operations that can be performed on them. The operations are created with a similar Create function, deleted with a similar Delete function, changed with a Change function, displayed with a similar Display function, and worked on as a group of like objects with a "Work with" function. Most types of objects can be renamed, moved, saved, or restored using one set of commands which operate on multiple object types. Therefore when users are presented with a new type of object, they can expect the new object to have a similar design and behavior to those with which they are already familiar. This function predictability makes it possible for users to feel comfortable and in control.

Figure 9 shows the primary groups of objects, along with several examples of objects in each group. The attributes and operations of objects within each group have even greater similarity.

This object-oriented design allows a programmer to define workstation and printer devices to the system, create files, create application programs, and create job-processing environments in a convenient, straightforward fashion. It gives the flexibility and extendability needed to allow the system and applications to be defined to meet each installation's needs.

Standard versions of all objects necessary for an operational system are shipped with the system. The initial or small-system user does not need to create objects, such as job queues or output queues, to get started. The extendability and flexibility are available when needed.

Object-action flow. There are two basic methods by which users request work. One method is to first identify the object to be acted on, followed by the action to be performed on the object (object–action). A second method is to first identify the action desired, then the object on which to perform the action (action–object). The flow for each of these two approaches is shown in Figure 10. Both are likely methods for users to approach different tasks, and both are supported on the AS/400.

The object-action approach provides many productivity and ease-of-use benefits. It is supported by having the user first identify the type of object via a menu, after which a "Work with" list panel of the objects of that type is presented. The "Work with" list panel is the cornerstone of this approach (see Figure 5). On it the user may directly and repeatedly request actions on the listed objects via action options. Entry panels are shown when an action requires qualification.

Some of the advantages of object-action approach are:

- It allows seeing the objects before deciding which actions are needed. The name as well as key attributes are shown in the list entry for each object. The user can display the contents of the object.
- It allows the user to conveniently change to a different action after seeing the object, without leaving the current panel showing the object or the list of objects.
- It allows the user to point to the name of the object rather than having to type it correctly from memory. (The action-object approach can also provide this through the prompt dialog action, but it requires an extra step.)

Figure 9 Object type groups

Group	Object type/subtype	Content
File	File Physical database file Logical database file Display file Printer file Intersystem Communications Function (ICF) file	Data and data description
Program	Program Control language program RPG program COBOL program	Processing description
Description	Device description Line description Subsystem description Job description	Set of attributes
Queue	Job queue Output queue Message queue	Waiting line
List	Authorization list Configuration list Document list	List of entries
Directories	Library Folder	Directory of objects
Other	Document Graphics symbol set Query definition	Application-specific

- It supports the convenience of performing an action on multiple objects.
- It supports performing multiple actions on multiple objects in a very natural way.
- It lends itself to performing a sequence of actions, one after another, without leaving the list.
- The object-action path is normally shorter because the panel of the list of objects is reshown after each set of actions is performed, allowing one or more additional action requests to be entered. With the action-object approach, the flow returns the user to the menu, and the entire sequence must be followed again for each action, including at least one entry panel and optionally a selection list to identify the object (see Figure 10).

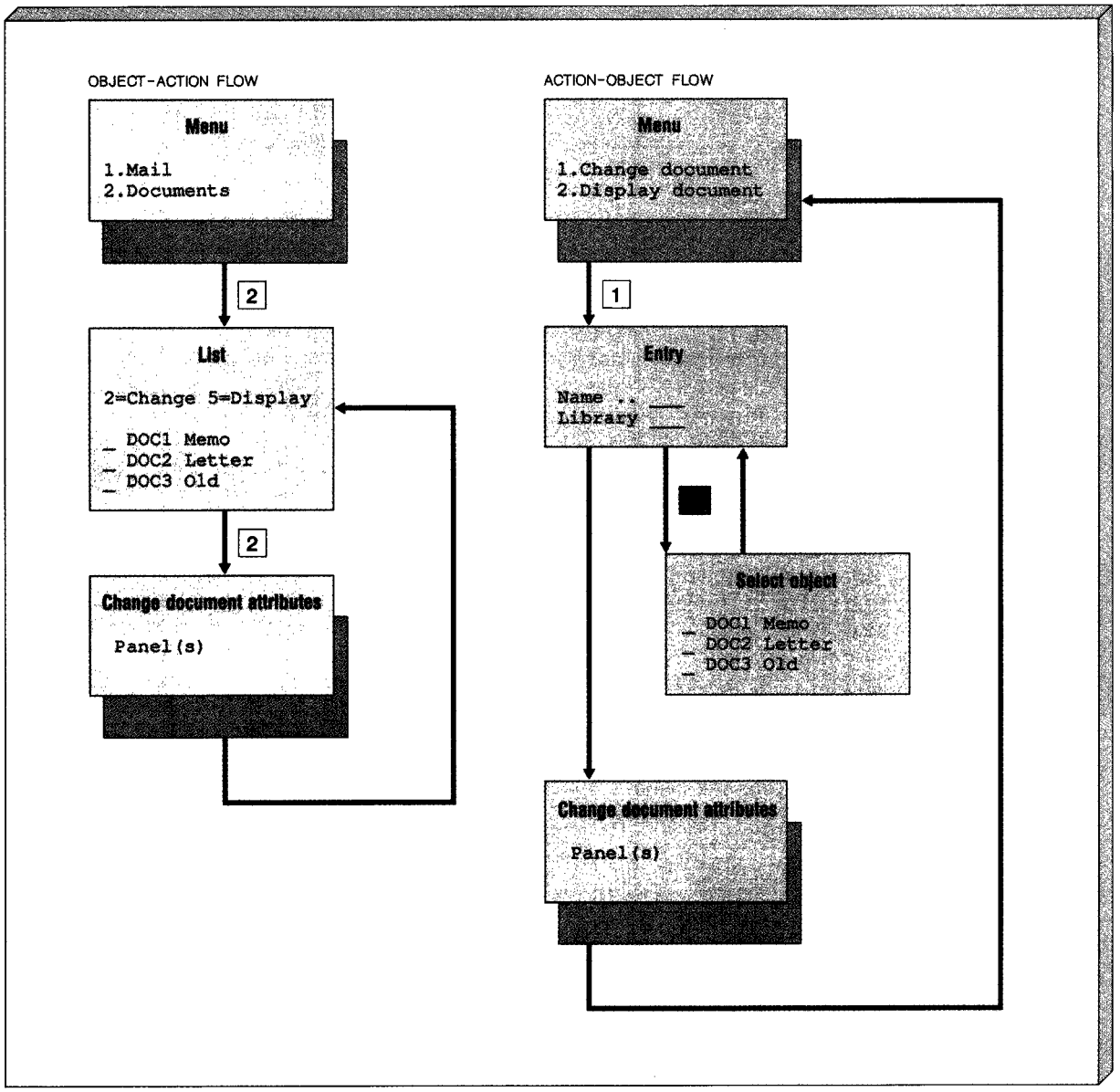
The action-object approach is supported by having primary tasks as menu choices in addition to the work with object type choices. After a task is selected, an entry panel is presented for the user to type the name of the object and any other options. The user may be given the opportunity to request a list of objects from which a selection is made (see Figure 6), using the prompt common dialog action (the F4 key), thus avoiding the necessity of having to key a name from memory. The action being performed cannot be changed at this point; the user can only select an object for the task in progress.

Recognition and selection. Another pervasive design approach that is used in the AS/400 is recognition and

selection. Recognition is easier than recall; selection is easier than keying. Wherever possible, users are presented with a list of choices and allowed to make a selection, rather than having to remember a name or command. Menus are one of the methods used to implement this design approach, listing actions and object types (or high-level groupings) that allow the user to select by number.

The "Work with" list panel discussed previously is another major application of this technique (see Figure 5). First, the objects are listed along with other information to assist recognition, such as date, status indication, and text description. Second, the supported actions are listed across the top of the list panel along with the assigned numeric action options (e.g., 2=Change, 3=Copy, 4=Delete, 5=Display)

Figure 10 Object-oriented flows for change document attributes request



used in selecting one of them. The user needs only to key an action option next to the desired object and to identify the object and the action to be performed on it. No keying of the object name or attributes is required. The action option simultaneously selects the object and the desired action.

On entry panels, recognition is used by showing the list of choices to the right of each entry field having a small set of choices. For example:

```
Type style . . . __  1=Bold, 2=Elite, 3=Pica
```

If the set of choices is long or varies in length, as does a list of document names, the prompt dialog action (F4) may be requested to provide a list panel showing a list of choices and allowing recognition and selection.

Interface layering. Layering is an approach that is used to address the reality that only a relatively small portion of the function is commonly used. The remainder of the function addresses special-case situations. This special-case function can detract from the simplicity and flow for average users and can make the product difficult to learn for new users. Layering simplifies the flow for new and average users, while still providing good support for the special situations. The dialog is structured so the special function is never more than a keystroke away, but otherwise the details are hidden. This reduces the chances of the special function being mistakenly selected or confusing users.

One of the ways this approach is applied on the AS/400 is in the structuring of menus. Commonly used actions are assigned to the primary or low option numbers so they are clear and visible. Special-case options are grouped under a single option following the others. For example, on the AS/400 Main Menu, problem-handling options are grouped under a single option of that name, after the primary options. The Diskette menu has the following primary options:

1. Display diskette information
2. Format a diskette
3. Print contents of a diskette
4. Save
5. Restore

Two secondary grouping options are provided that allow access to less commonly used groups of options:

50. System/36 diskette procedures
70. Related commands

On many entry panels, layering is implemented by providing a function key (the F10 key) for requesting advanced functions. This means the user gets a simple entry panel that only presents the required and frequently used options. In the case of a Copy File function, the first panel prompts for the file name, the file to copy to, whether to add on the end or replace the information there, and whether to create a new file (see Figure 11). The experienced user can press function key F10 to get such advanced function prompts as a specific record format, offset record number, and whether to print a log of copied records.

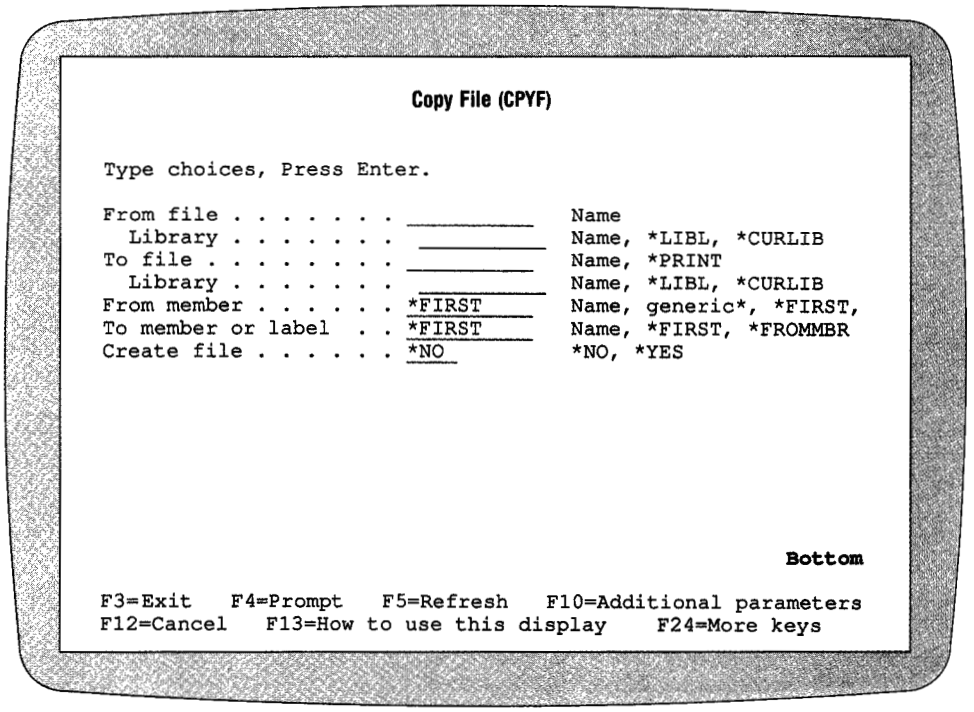
Grouping helps the user associate like items.

On list panels, layering is used in two ways: layering of the columns of information and layering of the actions supported. The columns of information shown initially are the ones that meet usual needs. If there is a logical grouping of additional columns of information available, the F11 function key is supported to request that alternate view.

The second use of layering on list panels is in the presentation of the action options in the instruction area. The initial set of action options shown is the set of commonly used actions; another set of less commonly used options is often available by pressing function key "F23=More options." The layering is done independent of the numbering of the options so that consistency in numbering for similar actions can be maintained across list panels, regardless of whether they are normal or advanced actions.

The same approach is used for function keys in the function key area using "F24=More keys" to access the other groups of active function keys.

Figure 11 Example of entry panel prompting for Copy File command



Grouping. By relating like pieces of information or functions and grouping them, it is possible to make the interface simpler to understand. Grouping helps the user associate like items and learn them collectively rather than individually. It is the process of subsetting a larger whole so that users can remember the groups, which are fewer in number.

Two ways to communicate grouping are alignment and separation. Both are used throughout the AS/400 interactive interface to communicate relationships, clarify function, and facilitate scanning.

Examples of the use of grouping on the AS/400 are the following:

- On entry panels the prompts, entry fields, and descriptive text stand out as distinct from each other because of being aligned and in separate columns.
- Blank separator lines are used to separate contextual information, instructional information, data, and function key descriptions.
- Within the panel body area, groups of entry fields or menu options are aligned and separated by blank lines, so they stand out and can be quickly scanned.
- Information on printed listings is grouped under subsection headings rather than being placed in a single continuous listing.
- The command set (over 800 commands) uses about 75 command verbs. The verbs form groups of like commands. Ten of these verbs or groups account for approximately two-thirds of the commands.

Fast paths. Fast paths are another design approach used to support users of different skill levels. Layering helps the novice and occasional user, but fast paths assist the experienced user. Fast paths may take one of two forms: (1) a shortcut for handling a portion of the current task, or (2) an alternative way to navigate to the application or task support.

A shortcut should not require a separate step to get to it. For example, a list panel with an input-capable top list entry allows users to key the name of the object, rather than scroll to it.

Another example of a shortcut is the parameter line on some "Work with" panels which allows the entry

panel to be bypassed for special qualifiers on individual actions. When an option is keyed next to a list entry that normally results in an entry panel to finish the specification, the values can be keyed into the "Parameters or command" field in command syntax to bypass the entry panel. (See Figure 5.)

Another example is a "Do all the above" option on a menu like the Display Job menu that allows the selection of a whole group of attributes to display without having to request each individually.

An alternative navigational fast path is a way to request a task which is based on where users are in their session.

One example is being able to request a task through the use of a command without having to go to the menu supporting it. While the interactive interface of AS/400 is carefully designed not to require a knowledge of commands, most actions result in a command being processed. These commands can be entered directly, which sometimes may be a faster method to request a function. The AS/400 commands match the function, terminology, and choices shown on the panels. This, coupled with the availability of a command line on most menus and list panels (see Figures 4 and 5), makes it very easy for users to begin using the commands for frequently requested functions. The same entry panels that are presented if a function is selected by number from a menu or list panel can be requested by pressing F4 while typing the command parameters. Any parameters already typed are carried over and filled in on the entry panels. Defaults are shown for any entry fields for which values have not been specified as parameters.

Another example of an alternative fast path is being able to go directly to a menu by name rather than by going through a sequence of menus. The GO command supports this option.

Still another alternative is the ability to interrupt a current task to do something else, without having to exit and return. The System Request (Sys Rq) key is active at all times, allowing users to interrupt their current task and perform one of several supported functions such as send a message, check on output status, receive messages, or start a separate session.

Keeping the user in context. For users to know where they are and feel comfortable, it is important to provide needed confirmation of "where am I" information. This is done in several ways:

- The title of the panel corresponds to the text of the selected action, whether it was a menu option, list panel action, or a function key.
- The panel ID in the upper left corner of a menu corresponds to the name that can be used on the GO command to get to the menu.
- When appropriate, identifying information is shown below the title in an information area, giving the source of the information and any criteria in effect. For example, for a display of the content of a particular output queue, the name of the queue is shown:

Printer Output Files

Queue: ELKGROVE

Forgiving. When errors are detected, the user needs a consistent, easy way to correct the errors, back up, or exit. Users should not be put in a position where recovery is impossible or seems impossible. Errors are considered normal occurrences, and the system treats them as such. Instructions on how to correct a problem are provided for each message. These are available by pressing the Help key for the message.

When an error is detected on an AS/400 entry panel, the values in error are reshown in reverse video, the cursor is positioned at the first value in error, and an error message is displayed. The reverse video allows rapid identification of errors, whereas cursor positioning allows easy correction. The message gives a description of the error. Additional information about the error can be requested by pressing the Help key with the cursor on the message. The user can change any of the input values and then press Enter to have them checked again. The F3 key can be pressed to exit.

Wherever possible, arbitrary syntax rules or requirements on the order of specification are avoided. Examples of this on the AS/400 are: tolerance of uppercase and lowercase letters for names where the distinction has no value; tolerance of the presence or absence of leading zeros for a numeric value; acceptance of synonyms and abbreviations when entering search words within the Search index capability of help; and allowing either the presence or absence of apostrophes around a character string with no embedded blanks. Distinctions in these cases would be viewed by users as unnecessary and frustrating.

Contextual help with word search. Even with a flexible, layered user interface, the time comes when a user does not understand how to use a panel or how to get started on a task. Through the AS/400 help facility, support information is immediately at hand.

The help facility provides both context-sensitive help (based on cursor position) and a searchable index of help topics. All help information is stored in the form of small building blocks, called information modules. As shown by path one in Figure 12, when a user presses a Help key, the help facility displays the information module associated with the area where the cursor is currently positioned. When the cursor is in other, nonspecific areas of a panel, extended help is provided, as shown by path two. The extended help consists of information modules on the use of the display as a whole, in addition to all of the field help modules describing the use of the individual fields. The modules are linked together so that users can move forward and backward through them to see all of the help for the panel. If users initially receive help for a specific field, they can get the extended help by pressing a function key (F2).

If users want help on another task, a concept, or a term that they do not understand, they can ask for the information by keying a request in their own words. They can request Search index from anywhere in help by pressing function key F11 and then entering search words or a sentence request. Each of the words (except words used as simple connectors such as "the" or "of") is matched against tables of keywords and synonyms, and a list of the topics that best match the user-entered words is displayed. In the example shown in Figure 12, the user has entered "move words." The search process finds matches for "moving" and "positioning" (which are synonyms for "move") and matches for "text" and "lines" (which are synonyms for "words"). As a result, the user is presented with a list of topics on moving text and positioning lines. The user can then select one or more of the listed topics, and their information modules are shown.

For more information see "The Application System/400 help facility—design philosophy and considerations,"[5] also in this issue.

AS/400 control language

A single set of commands, called the AS/400 control language (CL), allows experienced interactive users and programmers to request system, utility, and application functions, using consistent syntax and semantics. It includes commands for requesting any system function, some of which are for user job and output management, system operation, configuration, query, programming, and security management. Almost all commands can be used interactively, one at a time, or in a batch job. The control language is, in many ways, a high-level language for performing system functions and application control. It can be compiled for more efficient performance. In a compiled program it supports variables of three data types, character, decimal, and logical; full-screen input and output to the display workstation; arithmetic functions; and calls to applications written in any language.

A Create Command function is provided to allow users to create their own commands or their own versions of system commands. These commands can invoke CL or high-level language programs, thus allowing applications to benefit from all the capabilities of CL including prompting, defaulting, and validity checking.

Syntax. The basic syntax of the CL is simple and free form. The blank is the separator character. It is a natural separator that is common to all countries. The period and comma are supported as decimal point characters, allowing worldwide portability and a natural syntax. The command name and associated parameters can begin in any character position, thus allowing indentation and parameter alignment. Each parameter has an associated keyword that can be used to identify the parameter value. The keywords may be omitted for the first set of parameters as long as the values are entered in the fixed positional order. For example, the Copy File command is defined to have the following keyword form:

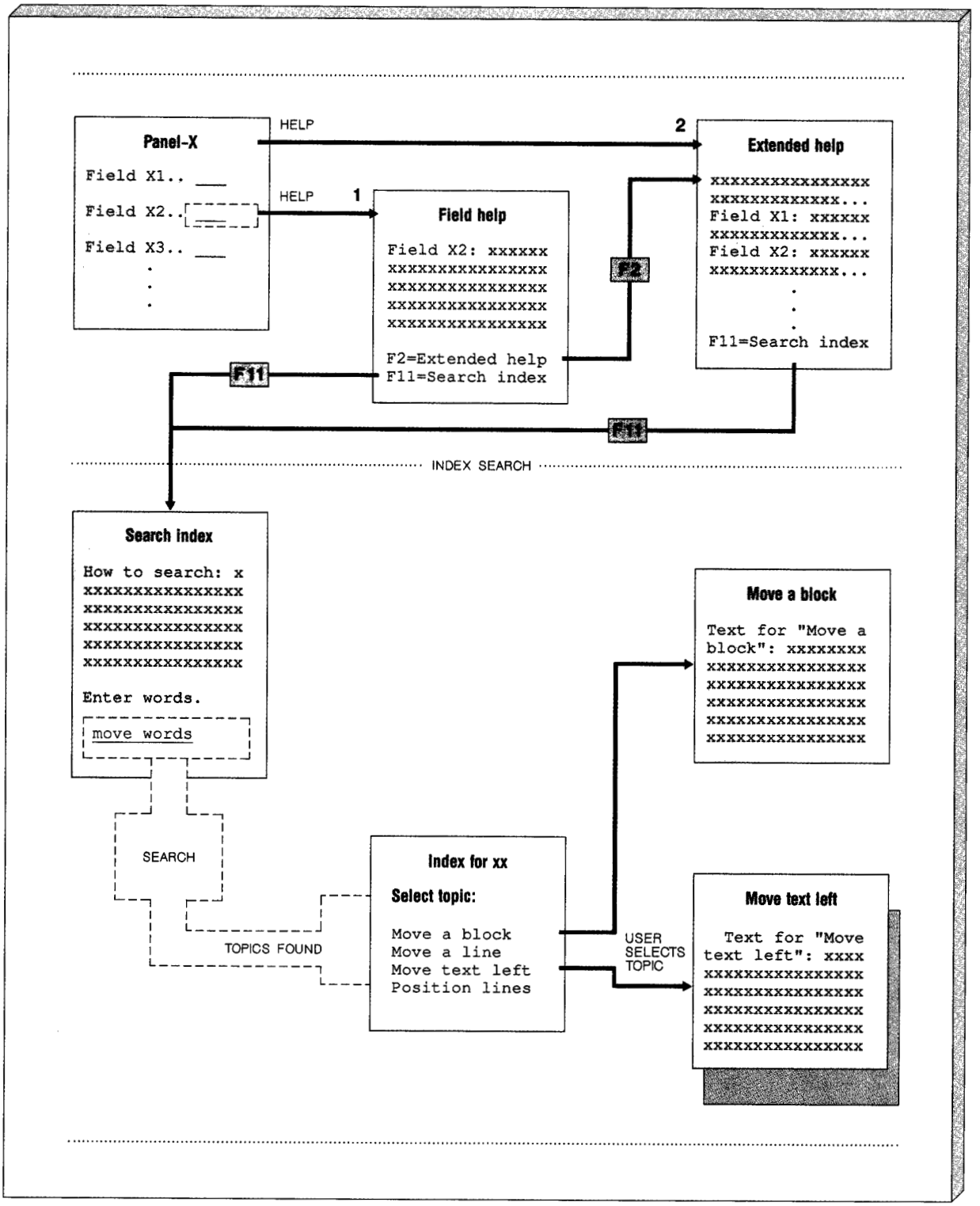
```
CPYF FROMFILE(file name) TOFILE(file name) . . .
```

With the keywords coded, the values can be coded in any order. A request to copy file A to file B can be coded with keywords in either of the following two ways:

```
CPY FROMFILE(A) TOFILE(B)
CPYF TOFILE(B) FROMFILE(A)
```

The same request can be coded positionally without keywords, but then the values must be coded in the order defined in the command definition.

Figure 12 How a user gets help—contextual and word search



For example:

```
CPYF A B
```

Command naming conventions. By convention, each system and utility command is designed to request a single function. This is normally an action-object pair. No matter what logic path (object-action or action-object) is used to narrow a request, the simplest and thus final request level is an action-object pair. This results in a very simple command name, action object, and a focused set of parameters which directly pertain to the request. This approach reduces the complexity of the command by reducing the number of conditional parameters. For example, if the command performed an action on multiple types of objects, different parameters may be needed for the uniqueness of each object type, adding significantly to the difficulty of learning and use.

The command set is then designed based on a single set of action verbs, some of which are: Create, Change, Delete, Display, Work with, and Copy, and a set of objects, some of which are: File, Program, Device Description, User Profile, and Job. The action verbs create noticeable groupings of commands, greatly simplifying the command set that numbers more than 800 commands. Some of the commands are the following:

- Create User Profile
- Change User Profile
- Display Device Description
- Display Job
- Copy File
- Work with Active Jobs

In order to simplify keying, the command names are abbreviated using a fixed-length abbreviation scheme of concatenating the individual abbreviations of each word in the name. Abbreviations, other than the ending one, are formed using the first character of the word, followed by the two most prominent consonants. Three characters allow for sufficient and meaningful uniqueness, even for similar words like Rename (RNM), Remove (RMV), Replace (RPL), Restore (RST), and Receive (RCV). The use of consonants maximizes uniqueness and predictability and avoids forming an inappropriate non-English word. This entire approach results in a very predictable, consistent command set that offers additional advantages over "minimum length," "minimum truncation," and a "single abbreviation for the whole command" schemes. These are:

Table 1 Command abbreviation construction

Commands	Abbreviations	
	Individual	Combined
Create User Profile	CRT USR PRF	= CRTUSRPRF
Change User Profile	CHG USR PRF	= CHGUSRPRF
Display Device Description	DSP DEV D	= DSPDEVD
Display Job	DSP JOB	= DSPJOB
Copy File	CPY F	= CPYF
Work with Active Jobs	WRK ACT JOB	= WRKACTJOB

- Less learning is necessary; all commands are built from the same abbreviations.
- A command's abbreviation is always valid, not being dependent on the environment or release to decide how many characters are necessary to determine uniqueness.
- Users can predict the name of the command they need by just knowing the naming rule.
- Users can parse any command name by only knowing the three-character rule, and therefore determine the command's function.

The last abbreviation in a name is sometimes reduced to less than three characters to minimize keying. This is done only if it can be done consistently and does not produce any ambiguity in parsing the abbreviations. For example, D is used for Description, and F for File, both of which identify a class of object and appear at the end of command names. In a few cases, exceptions are made to the vowel rule due to the strong precedence of common-use abbreviations that themselves were already three characters. Examples are LIB for library, DEV for device, and JOB for job. Common acronyms are used for multiple word object names, like RJE for Remote Job Entry.

The command abbreviations for the set of example commands are shown in Table 1. The approach has proven to be very extendable and rememberable. Users have found that they can readily construct or learn the names because of the strict consistency and the intuitive action-object naming.

Keyword and value naming. A single, unabbreviated word is preferred for keyword and value names in order to make them easier to remember. For example, some common keywords are FILE, TYPE, OUTPUT, and TEXT. Some common special values are *YES, *NO, *ALL, and *NONE. If the keyword or value name

represents a multiple word phrase, like user profile, the rule scheme for concatenating three-character abbreviations is used (for example, USRPRF). In either case, the values are defined so as to be self-documenting and not ambiguous. Values may be numeric, character strings, names, or special values.

Special values are the identification of predefined options within a keyword parameter. They are always preceded by an asterisk to distinguish them from the name of an object. For example, the FILE keyword for the Display File Description command has a special value of *ALL representing the option of listing the description of all the files, not just one. This can be shown as:

```
FILE(*ALL or file name)
```

The file name specified could be any file name, including ALL.

This approach to naming results in consistent self-documenting keywords and values without having reserved name restrictions that would be error prone.

Validity checking. Early validity checking is a system-wide strategy and occurs no matter where commands are entered. It is performed when a command is entered on a command line, during interactive command prompting as individual groups of parameters are entered, at source entry time as commands are entered into a database file for later compilation into a CL program, and at compilation of a CL program. A job option exists to have the CL commands in a batch job validity-checked as the job is placed on the job queue for later batch execution.

The validity checking is defined when the command is described to the system through the Create Command function. This results in a command description object which contains the full description of the command necessary to do a thorough validity check. The validity checking is done by a common command analyzer, no matter when and how the command is entered. This assures consistency of error identification.

Parameter defaulting. The AS/400 control language utilizes a highly visible defaulting approach. Most parameters are optional. Each optional parameter is displayed with a carefully selected default. Using an approach with many defaults helps the user by requiring less keying and less system knowledge. Less knowledge is required because not all parameters need to be understood to perform the desired task. Parameters can be left to default to get the function up and running. As learning progresses, values can be specified to meet special needs. For example, an output queue can be created with the number of job separators defaulted to one. Later, when it is found that three would be better, it can be changed.

A system command prompter is available to assist in entering commands.

Many systems have used some form of defaulting approach but have suffered from a number of common problems.

One common problem is that of not making defaults visible or what is expected. On the AS/400, the mystery is avoided by making all defaulting visible. The default value is always one of the standard user-specifiable values, and the capability is given to change defaults. Each value is fully described in the supporting documentation.

On systems where the default is not required to be a standard value, the default action tends to develop its own idiosyncrasies, such that letting a parameter default results in several unexpected ramifications.

Parameter prompting. On many systems, interactive command prompting is often inadequate or not provided. Some systems do not have any help behind the prompting. On still others, the prompting is only available after the user makes an error.

As part of the AS/400 interactive interface, a system command prompter is available to assist in entering commands. Prompting can be requested at almost any time while entering the command. Users can key whatever is known and ask for prompting assistance for the remaining parameters by pressing the F4 (Prompt) key. In this way the interface adapts to match the user level. The assistance is available for

the user needing it, but is not a frustration to the user not needing it. The same prompting is provided when entering a command interactively for immediate execution or entering a command as part of a CL program for later use. The prompter presents entry panels that identify parameters, defaults, and valid values so that the user can enter commands without frequent reference to publications. The list of valid values for a parameter can be requested by pressing F4 while in the field in question. Help text can be requested for the parameter by pressing the Help key.

The prompting is layered so users are initially prompted only for required and frequently used parameters. Parameters that are less frequently used are not initially shown. They can be requested by pressing F10 (Additional parameters) and thus be shown. Parameters that are only required in some situations are also not initially presented. They are presented on a following panel if it is determined, based on the initial responses, that more parameters are indeed necessary. For example, if a copy request references a diskette file, diskette-related parameters are shown but tape options are not shown. This is called "intelligent prompting." The system tailors the prompts based on user responses.

Prompting frees users from specifying parameter names or syntactical delimiters. They see only text prompts and blanks for entering individual values. Users can quickly review the command parameters and their defaults and key only the values that they want to change. When the Enter key is pressed, the values are validity-checked. If any errors are found, the panel is reshown with the values in error in reverse video with appropriate error messages. The cursor is positioned on the first value in error, and the keyboard is unlocked for easy correction of the value.

The prompting support exhibits many of the capabilities built into the CL: consistency, defaulting, and validity checking. Both prompting and the CL in general have received high user-satisfaction ratings on surveys of users.

Summary

With the introduction of the AS/400 comes an advanced nonprogrammable terminal user interface that spans the comprehensive facilities of this midrange system. The interface is designed to be used by a broad spectrum of users and provides them with interface capabilities not previously available to users of general-purpose computers. The interface is designed to allow each user to comfortably begin work and grow in productivity, using menus, layered entry panels, list panels, and command lines. It is a very consistent interface across all AS/400 function and introduces consistency between programmable workstations and nonprogrammable terminals, without compromising the potential and strengths of either.

A complete interactive interface is provided for all system-supplied functions including, for example, end-user, operation, programming, office, query, and problem-determination functions. A User Tasks menu is provided to meet the needs of end users. It and its accompanying facilities support displaying messages, sending a message, submitting and controlling jobs, checking on and changing output, and signing off in a way designed for end users.

An improved list panel is used to simplify creating and working with objects. It allows actions to be performed directly on the objects in the list or by typing the name. This allows actions to be performed consecutively and in groups from within the list area, simplifying and streamlining work.

Whenever an action is requested that requires additional information, an entry panel is provided that overlays the interactive specification of option. The frequently used options are presented first, and then, on request, more advanced options are displayed. Options whose applicability depends on other responses are only presented if appropriate.

At any time, users can ask for help that corresponds directly to their needs. In addition to presenting information describing the current field, the help facility allows users to request other information by typing words describing what they want to know. In response, their request is satisfied by the presentation of a list of topics from which they can choose the ones they want displayed.

Because the interface is based on CUA, its capabilities can continue to be extended to use other CUA interaction techniques and to make greater use of programmable workstation capabilities, preserving a level of consistency with CUA-complying products and today's AS/400 interface.

The single control language allows appropriate persons to install, configure, operate, test, and define

applications through a single set of commands. All the ease-of-use of command prompting and validity checking is available to them as well as to any application. Programs can be written directly in CL and make use of system functions or individual CL commands, and can be executed directly out of high-level language programs.

Feedback from a large number of customers verifies that the user interface of the AS/400 has made significant strides. The comprehensive interactive interface, which is designed for those who are not data processing professionals, and the underlying control language are a powerful and flexible combination. As the use of computers increases and more and more users who are not data processing professionals join the ranks of computer users, users' expectations and the need for greater ease of use will continue to rise. Therefore, completely new approaches and refinements in current user interface design approaches must be developed in order to continue to make computers more valuable to their users.

Acknowledgments

The author would like to acknowledge Dr. Joe DiCecco for his significant contribution to the AS/400 user interface design. He was the primary software human factors engineer. In addition, Dennis Charland and John Harrington were the user interface designers responsible for the help function design and the user interface standards, and served as key reviewers for this paper as well.

Cited references and note

1. In this paper, the term *nonprogrammable terminal* refers to keyboard-display devices attached to a host processor in which all or most of the user-interface functions are controlled by the host. The term *programmable workstation* refers to keyboard-display devices in which all or most of the user-interface function is controlled by the workstation itself.
2. *Systems Application Architecture, Common User Access: Panel Design and User Interaction*, SC26-4351, IBM Corporation (December, 1987); available through IBM branch offices.
3. *Systems Application Architecture, Common User Access: Basic Interface Design Guide*, SC26-4583, IBM Corporation (available through IBM branch offices, fourth quarter 1989).
4. "Expanded role for the programmable workstation in Systems Application Architecture," IBM external announcement letter (April 17, 1989).
5. D. A. Charland, "The Application System/400 help facility—design philosophy and considerations," *IBM Systems Journal* **28,** No. 3, 424–442 (1989, this issue).

General references

"Defining AS/400 compatible displays using data description," *Specifications Newsletter*, GC21-8163, IBM Corporation (August, 1988); available through IBM branch offices.

R. E. Berry, "Common User Access—A consistent and usable human–computer interface for the SAA environments," *IBM Systems Journal* **27,** No. 3, 281–300 (1988).

J. H. Botterill, D. A. Charland, J. Y. Harrington, "An integrated user interface," *IBM Application System/400 Technology*, SA21-9540, IBM Corporation (June 1988); available through IBM branch offices.

J. H. Botterill, "The design rationale of the System/38 user interface," *IBM Systems Journal* **21,** No. 4, 384–423 (1982).

AS/400 System Operations: Display Station User's Guide, SC21-9744, IBM Corporation (June 1988); available through IBM branch offices.

AS/400 Programming: Control Language Programmer's Guide, SC21-8077, IBM Corporation (June 1988); available through IBM branch offices.

AS/400 Programming: Command Reference Summary, SC21-8076, IBM Corporation (October 1988); available through IBM branch offices.

AS/400 Programming: Security Concepts and Planning Guide SC21-8083, IBM Corporation (June 1988); available through IBM branch offices.

Systems Application Architecture, Common User Access: Advanced Interface Design Guide, SC26-4583, IBM Corporation (available through IBM branch offices, fourth quarter 1989).

J. Howard Botterill *IBM Application Business Systems, Highway 52 & NW 37th Street, Rochester, Minnesota 55901.* Mr. Botterill is a senior programmer in the Software Strategy, Architecture, and Planning group at IBM Rochester. He is responsible for the AS/400 user interface strategy. He joined IBM Rochester in 1967 and helped develop the Multiple Terminal Monitor Task (MTMT) terminal system for System/360. He worked on the Communication Control Program (CCP) for the System/3 and had the design control responsibility for the System/38 user interface. From 1982 to 1984, he worked at the System Products Division headquarters in White Plains, New York, coordinating the division's usability process. Since that time, he has worked on the design of the IBM Common User Access user interface and the design of the AS/400 interface. He received his B.S. in mathematics from Wheaton College, Wheaton, Illinois, and his M.S. in mathematics from the University of Michigan at Ann Arbor.

Reprint Order No. G321-5369.