Through a nationwide network of interactive terminals in a teleprocessing configuration, users perform over twenty major business functions by sharing a single large and varied data base.

Emphasized are system design principles of the central complex whereby terminal message processing and data-base management are independently yet cooperatively performed.

Also discussed is system security, which includes user authorization and data-base reconstruction and auditing.

A large-scale interactive administrative system

by J. H. Wimbrow

This paper discusses an interactive terminal-oriented teleprocessing system for performing many of the administrative operations of the IBM branch and regional offices. The main impetus for developing this system was the urgency for a more responsive customer order entry facility. Increasing complexity of systems available and increasing sales volume were straining the relatively advanced IBM 1410-based order-entry system in use prior to 1969. Outside of the company, the strain revealed itself in slow response of order and delivery-date confirmation and occasional inaccuracies of system configuration. Internally, this slow reaction time was no less disconcerting.

order entry

For these reasons, an advanced administrative system was conceived and built. That system now performs the order-entry operation (as well as some four-hundred fifty other logical transactions) interactively in real time rather than requiring days or weeks. Since the process of entering orders initially stimulated the administrative system development, we first give the following functional steps for order entry:

1. Order sent to the processing center
2. Ordered system checked for validity
3. Order expanded into its system components for manufacturing

4. Delivery data assigned and transmitted to the branch office
5. System component numbers recorded for inventory control
6. Order summarized for both sales and manufacturing management control

validity checking

The earlier order-entry system offered many opportunities for delay. Original orders and delivery date assignments were transmitted by mail. Further, a significant part of the order processing cycle was the validity check. Such a check was (and still is) made because the configuration of a computer system usually involves a number of system components, many of which are interdependent and have prerequisites. Validity checks determine that all prerequisite devices are present on original orders and that the configurations ordered can be manufactured and can run. Where the ordering system determines missing prerequisites or other incompatibilities in the original order, the ensuing manufacture might result in an unrunnable system. In the earlier ordering system, necessary order revisions further delayed the already lengthy order processing cycle. Such revisions also contributed to the management and inventory control difficulties of reconciling the order totals carried by the manufacturing plants and the on-order backlog carried by the sales division.

inventory control

Inventory and management control involves three files: (1) open orders (which is another way of looking at the order-entry process); (2) uninstalled inventory (equipment that has been manufactured, but has not been installed); and (3) installed inventory. Further, the installed inventory is the basis for issuing machine rental invoices to customers. In this process, accounts receivable debit entries are also created. In the earlier administrative system, there occurred occasionally a geographic mismatch between IBM payment receiving centers and customer paying centers. Such a mismatch might have occurred with an account that received invoices for payments that were to be collected in several geographic locations, but were paid at a central location. This problem has been eliminated by the planned administrative system.

advanced administrative system

In 1964, IBM was embarking on a great new venture in information processing systems. The anticipated introduction of the System/360 would increase almost exponentially the number of possible configurations available for ordering, and hence, magnify the complexity of the order-entry process. Projections indicated that system complexity coupled with a growing volume of orders were likely to stretch the existing order-processing systems almost to the breaking point. Added personnel and equipment to bolster the existing systems would ultimately have become inefficient. Such problems stimulated a study group that in 1965 recommended developing the new administrative system having the following characteristics:

- Operate interactively with administrative personnel
- Interconnect branch and regional offices, headquarters, and plants (320 geographic locations having a total of about 1500 terminals)
- Operate conveniently and easily
- Respond to orders and queries in personal-time scale

The system design phase was begun by surveying the existent systems that most nearly approximated the requirements of administrative support for the IBM branch office organization. The closest counterpart was the SABRE airline reservation system.[1] Developed in the late 1950s and early 1960s, SABRE taught us much about designing and implementing large-scale, terminal-oriented, interactive systems. Airline reservation systems are used primarily for seat sales and inventory control, wherein the inventory is large, complex, time critical, and widely distributed. Airline reservation systems demonstrate the feasibility of operating on a nationwide scale with a network consisting of hundreds of terminals interacting with a central complex by using the techniques of message queuing and conversational continuity.

conversational continuity

Conversational continuity is a technique by which a user may give one item of information to the system now and several minutes later enter a second item. The system associates the two items without dedicating itself to any one terminal during a message transmission. The terminals in both the reservation and administrative system appear to the user as though he were carrying on a continuous and exclusive conversation with the system, when in fact, there are hundreds of users carrying on simultaneous conversations. The airline reservation systems required specially tailored input/output terminals because standard interactive terminals did not exist when those operations began. Similarly, Central Processing Units (CPUs) had to be tailored to handle high volumes of teleprocessing input and large data bases. The administrative system, however, uses System/360, which is designed to accommodate high rates of teleprocessing activity and a large data base in standard system configurations.

administrative system application

To achieve the prime objectives of faster, more accurate order processing and the related internal operations of inventory control and accounts receivable, a certain system capability is required. That system, however, could handle a greater data processing load than was initially planned, and the number of transactions was incrementally increased until there are now about three-hundred fifty. Indicative of the branch office operations that the administrative system performs are the following applications:

- Order entry
- Delivery scheduling
- Territory assignment

- Payroll
- Commission accounting
- Configuration validation
- Accounts receivable cash application
- Customer master record
- Installed machines inventory
- Billing
- Customer student enrollment
- System user training (CAI)

CAI

Comparative studies were made between the interactive and batch modes of input/output and processing. Although batch order processing offered a lower cost per transaction, this saving was offset by longer response time and by higher user error rates and training costs. For user training, we drew upon the corporate research and development experience in computer assisted instruction. The system is designed to train and be operated by about 5,000 administrative users in the branch offices throughout the company. Classroom training of each user in only a few skills would entail an unacceptably large education program. By developing training courses for terminal instruction and by training the users at the terminals through which they enter and receive information, a substantial saving in training cost is achieved. There is also a significant improvement in user accuracy. Computer assisted instruction techniques also fit the overall logic of each of the applications because they are tree-structured as the training courses are.

standard user terminal

The IBM 2260 provides a terminal that further lends itself to tree-structure logic and a conversational mode. This alphanumeric terminal enables the system to be designed using the concept of the system's asking the user a specific question and giving him all valid responses. By depressing one key, the user selects the desired response. On the basis of human-factors considerations, this method is believed to be more efficient from the user's standpoint.

standard operating system

Another key factor that influenced system design was the System/360 Operating System (OS/360). Whereas the airlines reservations systems had found it necessary to develop their own operating systems, standard operating systems are a regular programming component of System/360. Although OS/360 and its options were expected to operate the administrative system, the release date of the required multitasking option was scheduled after the administrative system was to become operational. For this reason, it was necessary initially to have all real-time programs interface the real-time operating system (RTOS) developed by IBM for the National Aeronautics and Space Administration. When the multitasking (MVT) options became available in OS/360, a conversion was made from RTOS to OS/360 and the admin-

istrative system has run entirely on a standard OS/360 without modification since that time. Thus the projected system, termed the *Advanced Administrative System*, embodies the technologies and contributions of: (1) airline reservation systems; (2) a teleprocessing and large-storage oriented computer (using standard hardware); (3) alphanumeric display terminals; (4) computer assisted instruction; and (5) the standard operating system, OS/360.

system configuration

In this paper we present the administrative system design principles. Whereas system configuration and application programming details may be modified on the basis of experience and through continuing system studies, the basic principles discussed here have proved to be both sound and constant.

Consider the field terminal network (shown in Figure 1) and the central computer complex (shown in Figure 2) to constitute the overall system hardware configuration. The input/output components of the field terminal network are the approximately fifteen hundred IBM 2260 display terminals and IBM 1053 printers located in some three-hundred twenty branch offices, plants, and headquarters. Data are transmitted between these I/O devices and nine geographically distributed IBM System/360 Model 30 computers over low-speed telephone lines. These computers perform data concentration and retransmit data from the field to the central complex in White Plains, New York over high-speed telephone lines. The system is designed to respond to ninety-five percent of inputs in five seconds, and accommodate up to one and one-half million inputs per twelve-hour day (an average of fifty inputs per second). Programming for message processing and data management are the key topics to be discussed later in this paper. To give perspective to the overall system, Figure 2 shows the programming design reflected in the configuration of the central complex. We call this the "front-to-back" design. The front is shown consisting of three System/360 Model 65J computers that are used for message processing; the back is a Model 85K that performs data-management operations. Associated with both the front and the back system components are the required peripheral storage.

The spare processors shown in Figure 2 normally do off-line work. In the case of emergency, the spares are switched in automatically during a brief service interruption. The system uses an uninterruptable power supply to protect it from momentary power loss or to permit an orderly system shutdown in case of an extended power failure.

application programming

To give further perspective, Figure 3 illustrates the overall organization of the programming for the advanced administrative system. From a programming point of view, this paper empha-

Figure 1 Geographically distributed field terminal network

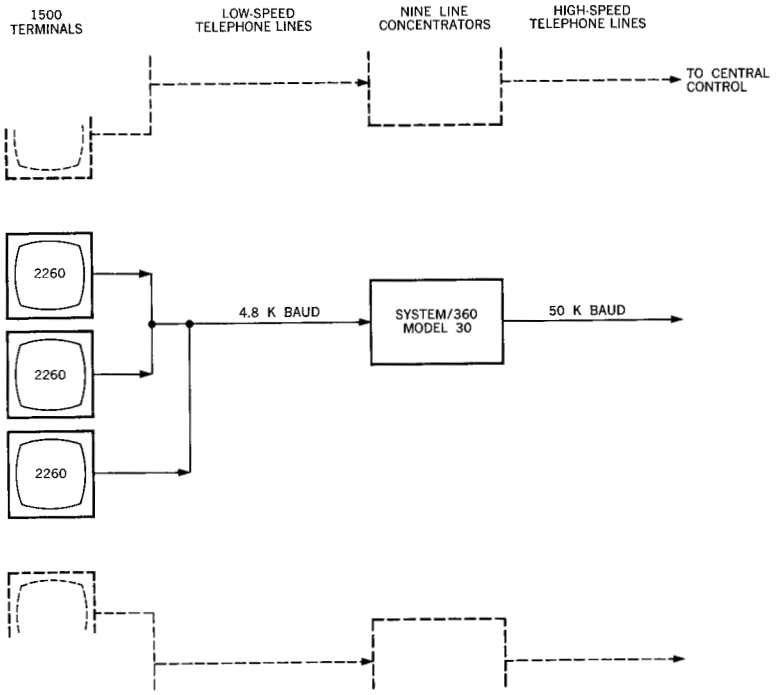
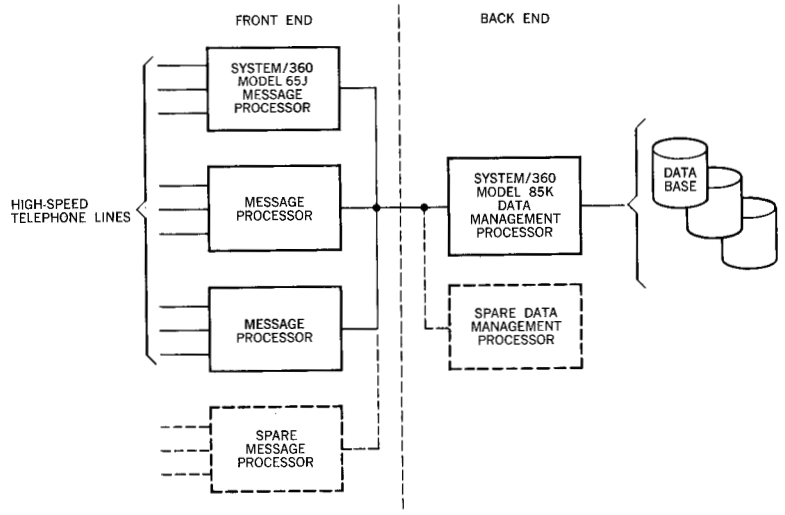
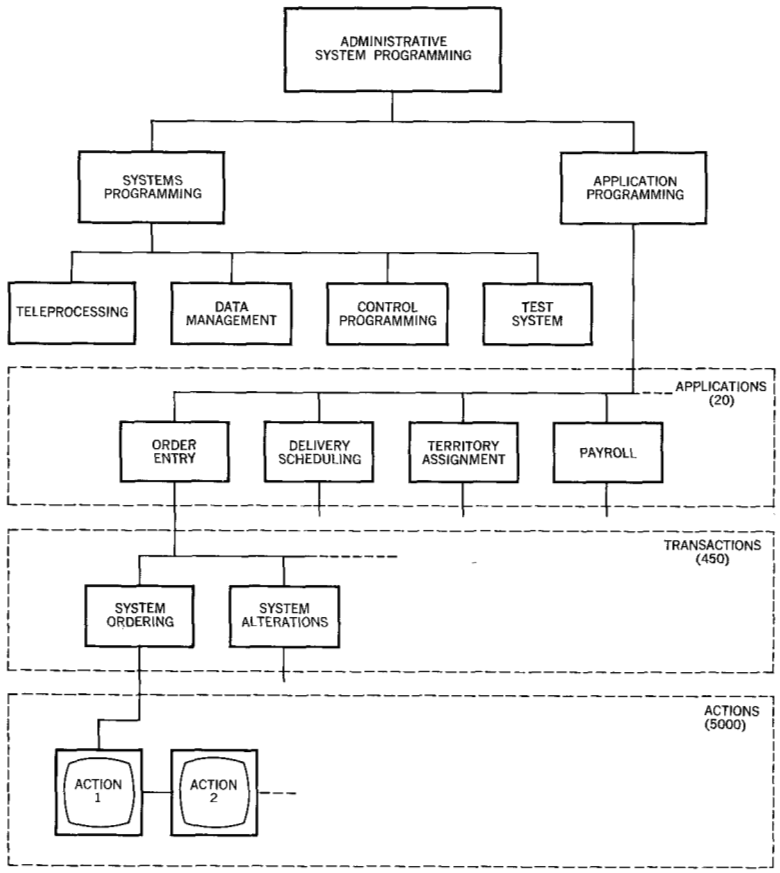


Figure 2 Administrative system control complex



sizes the control programming—especially message processing—and data management programming. These programs operate on individual applications programs, which are assumed programming units in our discussion. Therefore, we briefly summarize the programming organization here. In discussing design consid-

Figure 3 Organization of administrative system programming



erations earlier in this paper, an indicative listing is given of the administrative system applications. Figure 3 shows that there is an expanding capability of about twenty such basic applications. Many applications have several distinct phases or transactions. Therefore, a further subdivision of the applications are the approximately four-hundred fifty transactions indicated in Figure 3. For example, the order entry application consists of a system order program, a system alteration program, and several other order entry transactions. Similarly, the other applications may consist of several distinct transactions. These programs are further expanded into some five thousand possible action programs with which the user interacts at the terminal—for example a listing of possible colors available for a particular system being ordered. The action is the basic programming building block of the program organization.

front-to-back design

Analysis based on data collected early in the study phase indicated that more than one Central Processing Unit (CPU) would be required by the administrative system. When this requirement

had been determined, it was apparent that a technique would have to be developed to prevent two or more terminals serviced by different CPUs from attempting to update the same record at the same time without losing any of the updates. A number of potential solutions were evaluated.

One possibility was to divide the actions among the CPUs based on which files the CPUs accessed. Thus there would be no need to update the same file from different CPUs. Balancing the processing load under this type of operation would be difficult. A second possible solution was to set up a control mechanism to prevent one CPU from reading a record while any other CPU was updating the same record. This would have required elaborate processing logic.

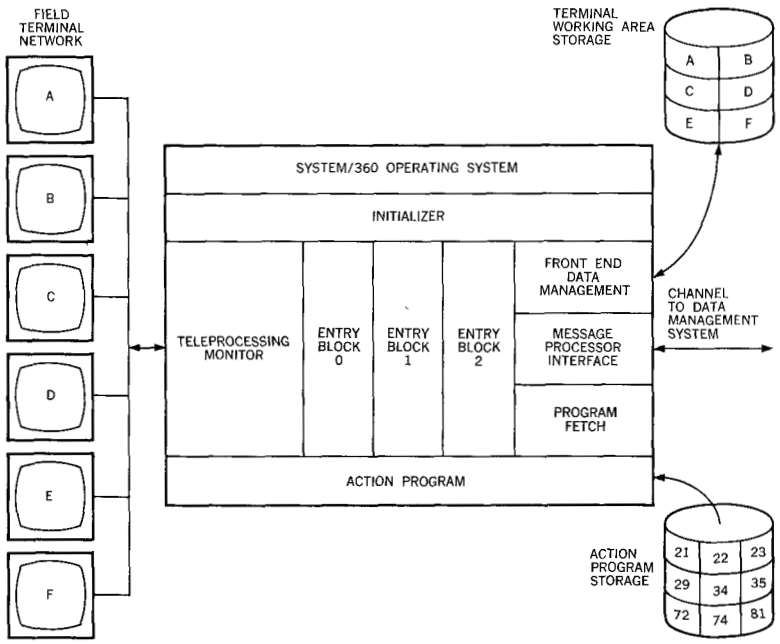
The approach adopted avoids the complexities inherent in either of these two solutions. The resultant front-to-back technique on which the advanced administrative system is based allows a reasonably simple control along with the ability to balance the processing load among the CPUs by dividing the work into message processing and data management. The design also provides for ready expansion as the load on the system grows. Message processors are added up to a practical limit of four, and this growth does not alter the basic architecture of the system. The work load of the message processors is balanced empirically by measuring their input loads each day and then switching the telephone lines each succeeding day to accommodate the measured loads.

By the front-to-back technique, the processing load is divided so that the inputs from all terminals and outputs to all terminals for all actions are handled by one or more message processors (front end). All file requests are developed by a message processor and become inputs from the front end to the data management processor (back end). Formatted data drawn from the files by a data management processor become responses to the requests of the message processor. Much of these same data in turn become the output of the message processor to the remote terminals. Thus, the message processors are connected to the telephone lines and process all messages from the terminals, and the data management processor handles all activity with the files. We now discuss the front-to-back design in terms of simplified models of the two basic processors—one message processor and the data management system.

Message processor

Consider the programming of the message processor of the advanced administrative system beginning with Figure 4, which

Figure 4 Message processor architecture



represents primarily main storage as allocated in the front-end System/360 Model 65J. (Figure 4 through Figure 6 have been simplified to highlight the flow of control and information.) The simplified system has six 2260 display terminals designated by the letters A, B, C, D, E, and F. Associated with each terminal is a peripheral storage area called *terminal working area* storage, which contains one record (A to F) for each terminal. If we added a seventh terminal G, we must add a record G to the terminal working area storage. The second peripheral storage contains the action programs for processing input messages from the terminals. Main storage is divided into areas for the operating system, initializer, teleprocessing monitor, and so forth as indicated in Figure 4.

Key to the message processor operation are the storage areas labeled "entry blocks." These working spaces are used for entering messages from the terminals, processing the information, and transmitting responses back to the terminals. The message processor in Figure 4 has six terminals, but entry blocks for serving only three terminals at a time. Thus the system must process messages rapidly so that it can process input messages from each terminal without causing a noticeable delay. Messages awaiting service queue up in the teleprocessing monitor. The application action program area in main storage holds several action programs that have been brought in from the action program peripheral storage as required by the input message.

Figure 5 Front-end message processing

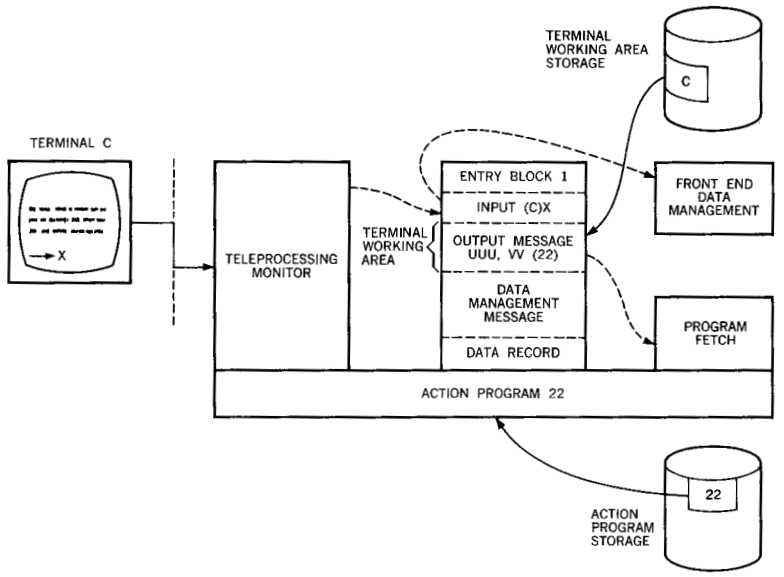
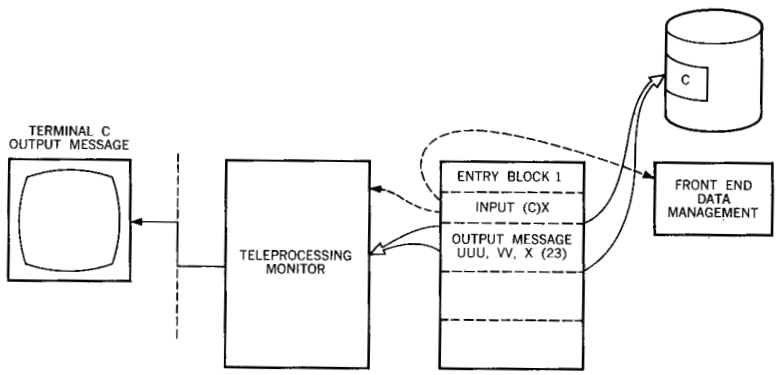


Figure 6 Output message



Consider now control and data flow initiated by an input message as shown in Figure 5. The entry block contains an input area and a terminal working area. The terminal working area consists of the display retention area, (which is the area where the action program stores the output message) and transaction continuity data (UUU,VV). Action programs perform all the logical operations necessary to service the user at the terminal. This may involve such a simple operation as asking the user the color of the machine he is ordering and accepting his reply. Similar to the output message is the data management message, which is, however, directed toward the data management system (back end). The data record is data returned from the data base.

Let us now follow a message through the system. Assume that terminal C is in the middle of a transaction and has been asked a question by the system. Such a question is represented in Figure 5: What color machine do you want? Enter X for red, Y for white, and Z for blue. The user keys in X.

The teleprocessing (TP) monitor receives the input message, which consists of the name of the terminal C and the input message X. The TP monitor queues the message C-X in an input buffer then determines the availability of one of the three entry blocks (0, 1, 2 in the simplified system). If an entry block is available the input message is stored there. If none of them are available the TP monitor holds that input message in its input queue. If the input queue becomes saturated with input messages the TP monitor stops polling the terminals. Thus the system may run out of time, but it cannot run out of storage space. The system is, however, designed to a five-second response time specification.

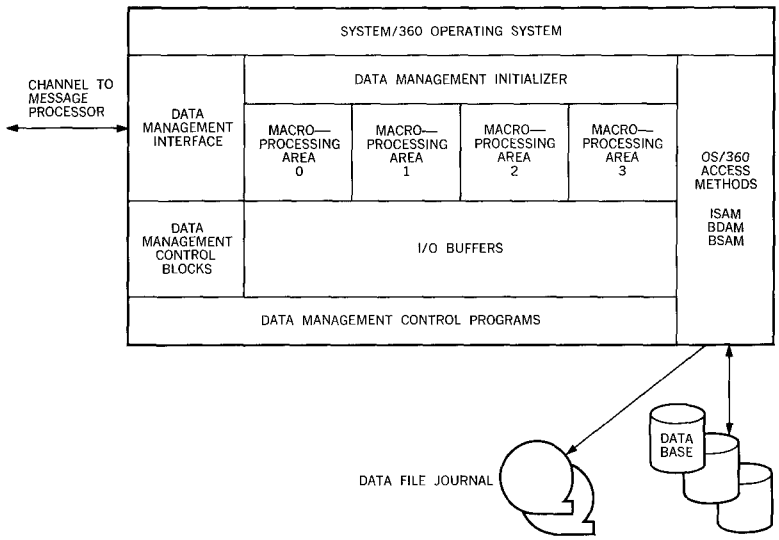
The TP monitor in our particular example, determines that entry block 1 is available. The TP monitor moves the name of the terminal C and the input message X into the input area of the entry block.

The TP monitor gives control to another systems program, called Front End Data Management (FEDM), which is signaled by the TP monitor that there is a new input message in entry block 1. The FEDM checks the input area of entry block 1 and finds that the message originates at terminal C. Since there is one record in terminal working area storage for each terminal, the FEDM reads record C from the terminal working area, and stores that record in the terminal working area of entry block 1. Figure 5 shows that terminal record C contains UUU and VV, which are the continuity of answers to previous questions in this machine order action. The answer to the current question to be added is X. Terminal working area record C contains a 22, which is the number of the program that is used to process the current input. We explain later how the action program number (22 in our example) appears in terminal record C.

Program FETCH—an OS/360 capability—reads a copy of program 22 into the action program area of main storage if it is not there. If action program 22 is already in main storage, FETCH gives control directly to 22. (All action programs are re-entrant.)

Action programs always do at least two things: (1) they transmit a message back to the terminal (C in the example), and (2) they name the program that is to handle the operator response to the message that is transmitted and displayed. (By the second step, program 22 appears in the terminal working area record C; it was put there by the process just described during the previous ques-

Figure 7 Data management system



tion-response interchange.) The message transmitted to the terminal is typically a question, and the action program named handles the operator response. In our example, action program 22 determines whether X is a valid response, which it is, and moves X into the main storage terminal working area for terminal C. In this way, valid responses are accumulated and the order is built up.

Having thus saved the response, program 22 may issue a request to the data management system by a data management request. The reason for the data management request in this example is to obtain the skeleton (format) of the output message, which is not a part of the action program. More extensive uses of the data base are discussed later in this paper. Control now transfers from the action program to another program designated as the message processor interface in Figure 4. That interface copies the data management request into its own buffer and returns control to the action program. The message processor continues processing terminal inputs and outputs independently of the data management system.

data management processor

The data management processor is the collective name for the system programs of the data management system (which also includes the data base to be discussed later in this paper). Figure 4 shows the message processor interface and Figure 7 shows the data management interface and both figures show the interconnecting channel. The message processor interface establishes communications through the channel to its counterpart in the data

management processor. The transfer of a data management request from the message processor interface to the data management interface initiates the retrieval of the record that action program 22 has requested. Action program 22 places itself in a wait state until the arrival of the requested record. The data management processor (which is discussed in more detail in the next section) resolves the indexing and blocking conventions used in the data base, and retrieves the requested record. The data management processor passes the record to the message processor via the channel and the interface monitors. The message processor stores the data record in the entry block that terminal C is currently using.

The action program 22 uses the display message skeleton transmitted from the data base to build the message to be transmitted back to the terminal C. The message to be transmitted is built in a display retention area of the terminal working area. This area is a part of the entry block, but is not specifically shown. Action program 22 now readies an output message for terminal C either asking a new question or transmitting a response to a previous question. By the rule previously given, program 22 must know the name of the program that will process responses to the message that is to be transmitted to terminal C. In the conversational mode, the user's reply cannot be one to a question that is not displayed; the design of the administrative system assures the user that all relevant questions pertaining to each application are displayed and answered. Thus we see in Figure 6 that action program 22 names action program 23 to process the user's answer to the current output message. When the output message for terminal C has been stored and the next program has been named, control returns to the TP monitor of the message processor.

The entry block contains the information just discussed—the output message and the name of the next program. The TP monitor reads that output information from the display retention area of the entry block and stores that information in the TP monitor output buffer. The TP monitor determines the terminal to which the output message is addressed because the output message is always transmitted to the same terminal that the input message came from. In the example, the output message is transmitted to terminal C.

Control is given to the FEDM as soon as the TP monitor has the output message in its own buffer. The FEDM now updates record C in terminal working area storage so that, in addition to UUU and VV, it also contains X. Record C in terminal working area storage also contains the name of the next program that will be used by terminal C, plus the message that has just been sent to terminal C in case retransmission is required.

A single terminal action has now been completed. The administrative system has been designed so that the previously discussed action generally requires less that five seconds, including TP delays, data management request delays, and queuing delays. The entry block previously used by terminal C is available for another message from any terminal that is transmitting a message to the administrative system. When the user at terminal C replies to the question in the previous output message, there is only one chance in three that the new response will use the same entry block because responses are assigned to the next of the three entry blocks available in the simplified system. More efficient use is made of system resources by the available assignment of entry blocks than would be attained if an entry block were dedicated to a single terminal until all of its transactions had been completed. This is true even though the terminal working area storage must be reaccessed to read terminal C data into another entry block. For the next action program 23 is called and the previously described steps are repeated until the entire transaction is finished. All action programs—about five thousand—have been designed to fit exactly the pattern described.

In the initial design of the advanced administrative system, all actions were dispatched as separate tasks under OS/360. Measurements made on that configuration during actual operations indicated that an excessive amount of CPU time was being consumed by task switching and resource management. An improved task dispatching design was conceived and was implemented in the system being discussed in this paper. The basis for the more efficient dispatching technique is the fact that all action programs function in exactly the same way under the system control programs. Therefore, the action programs are incorporated as subroutines of a single master task under OS/360.

This dispatching technique saves a large number of the task switches that were required in the earlier system design. Measurements indicate that CPU time for each action is reduced by more than fifty percent. Of equal significance is the fact no reprogramming of actions was required to make possible this programming redesign.

Data management

The data management system is the back end of the front-to-back administrative system, and consists logically of a control program (data management processor) and a data base. The data management processor has been discussed as it relates to the message processor. We now discuss the data management processor and the data base as they relate to each other.

processor

As shown in Figure 7, the data management processor is architecturally similar to the real-time message processor. Comparing the message processor in Figure 4 with the data management system in Figure 7, there are points of similarity and difference. The initializers, of course, perform similar functions. The macroprocessing areas are analogous to the entry blocks of the message processor because the overall functions of processing input messages or data requests and transmitting responses are similar in both cases. There are more macroprocessing areas than entry blocks because the typical action program in the message processor initiates six requests to data management. Some input messages, of course, make far greater demands than that.

The function of the data management processor is similar to that of the message processor. The data management processor receives its inputs from the message processor and responds to them on a one-input, one-output basis. In the case of data management, its input (from the message processor) is termed a data management request, and the output is in the form of a specific service. Such a service may be the delivery of a record or the notification that a record has been stored. All data management control programs are re-entrant, and they reside in data-management main storage at all times. The data management processor has no counterpart of the terminal working area storage for conversational purposes, nor are there any action programs in peripheral storage. Any information corresponding to these storage locations is in main storage at all times. The data management processor basically consists of the logical programs that drive the data base system, which is discussed in the next section.

The OS/360 access methods are used for the actual physical retrieval and reading and writing in the data base, but the data management control programs perform all the logic. These data management programs logically correspond to the application programs of the message processor, that is, the data management programs process the input messages to the data management system. Input messages to the data management system are put by the data management interface monitor in an available macroprocessing area, and control is given to a data management program to service each input message. Normally, such an input is a request for data from the data base. The data management program uses the information contained in a request—the input message stored in the macroprocessing area—to retrieve indexes from the data base that are used to locate individual data records in the data base.

data base

Before discussing the advanced administrative system data base, we first give an indication of its physical size in terms of the following statistics:

- More than 20 million data records

Table 1 Installed machine file

Record key (serial number)	*String key (system number)*	*Group key (customer number)*	*Description*	*Date of manufacture*	*Color*
1234	A9421	27123.00	2401 tape unit	xx/xx/xx	Blue
2345	A9421	27123.00	CPU	–	–
3456	A9421	27123.00	Card reader	–	–
0112	B0942	27123.00	–	–	–
0479	B0942	27123.00	–	–	–
4823	B0942	27123.00	–	–	–
7894	B0942	27123.00	–	–	–
3168	A9111	87941.00	–	–	–
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

- Approximately 27 million index records
- More than 2.5 billion bytes of data
- Approximately 0.5 billion bytes of indexes
- Indexes and data are stored on approximately 150 IBM 2316 disk packs on IBM 2314 disk storage devices

The data base is logically structured into individual records, strings, groups, and files.

A *record* is a series of logically related data fields pertaining to a given item. In Table 1, consider the example of a record for a specific IBM 2401 tape unit that is installed with a specific customer. The record contains all necessary information concerning that particular machine, i.e., description, date of manufacture, serial number, system number (number of the configuration of which the unit is a part), color, etc. All fields of that record are stored contiguously in the data base and are available for retrieval as a unit. All such records related to installed machines are physically located in the same file as the example record just cited. Thus we call the example file in Table 1 the installed machine file. The length of an individual record may vary widely from file to file. When, however, format and length have been established for a particular type of record, all records of that type conform to the standard. Each record has a unique key (called record key) that distinguishes it from all other records of the same type. In the installed machine file, the serial number is the record key – 1234 in the example of the 2401 tape unit.

Strings are collections of logically associated records in a file. Referring again to the 2401 tape unit in Table 1, it is convenient

to associate that unit with associated control units and other peripheral equipment used with the CPU, which together make up the computer configuration installed in a customer's location. Such a relationship is indicated by each of the individual machine records having the same system number, which is specified as the string key in the table. Thus, system number (or string key) A9421 consists of a tape unit (1234), CPU (2345), and card reader (3456).

Similarly, *groups* are associations of strings. Continuing to build on the example of the tape unit and system number, it is further convenient to relate all machines (record key) of all systems (string key) belonging to a specific customer (group key). Table 1 is a case in which the customer having group number 27123.00 has more than one system, wherein each system is made up of several machines. Relating all machines of all systems of a given customer constitutes a group relationship. In this manner, we can retrieve all individual machines for all systems for that customer. If the customer has several systems, that fact is specified by the system number (string key), and each component machine is specified by its serial number (record key).

Files, the highest level in the data-base hierarchy, are collections of identical types of records, which may also be associated by strings and groups. The example of the 2401 in Table 1, is a record in the installed machine file, wherein all records of installed machines have the same format and length, as mentioned before, and are considered as a single file.

The complete data base consists of approximately two hundred such files, all of which are organized as described above. A file may have up to three keys (record, string, and group). It is not necessary, however, to have three separate keys for all files. If a file does not logically break down into three levels of association, it may have only two levels of association, as in the accounts receivable file. Here, there is an individual record for every invoice issued. All invoice records are of the same length, and format, and they are identified uniquely by the invoice number, i.e., the record key. Since it is convenient to associate all of the invoices for a specific customer, a customer number is used as the string key. (There is no group key in the accounts receivable file because there is no need for a higher association.) Other files may have only a single-level key, i.e., the record key. Such a file is the one that specifies the production schedule for each individual type of machine manufactured. Since that schedule is not related to any higher association, a single key (record) suffices to identify the file.

All data records are written using the basic direct access method (BDAM), using the sequence of record key within string key, with-

in group key. As these records are written initially, BDAM supplies a feedback of the relative block number at which each record has been written. In Table 1, the machine whose serial number is 0479 is the fifth relative record in the BDAM data set. If BDAM is asked to retrieve the fifth relative record in this data set, it acquires the record that is identified as record key 0479.

The data management system must capture the fact that the record identified as 0479 is the fifth relative record and store that information. Storage for relative-address information is an index file, which itself is a series of small records. Thus, when a message requests record 0479, the index shows that it is the fifth relative record in the installed machine file. When it is requested to do so, BDAM retrieves the fifth record in that data set. In this manner, an index is built for each different type of key for each data file within the system. Therefore, data files with three keys (record, string, and group) have three index files to permit the retrieval of records from that file by each of the three possible keys.

There must be one index record for each data record to retrieve data by the record key. It is not necessary, however, to have a one-to-one correspondence between strings and groups and their respective indexes. String keys, as an example, take advantage of the physical arrangement of the file and only have indexes pointing to the first record in each string. Other data records of each string are thus retrieved by reading them sequentially. The same technique is used for group-key indexing and retrieval.

New records added to the file are placed in an overflow area specifically set aside for that purpose. Index records are created and/or modified to logically integrate the new records into their respective files. This is one of several housekeeping functions that are scheduled in a batch mode when the terminal network is shut down. From time to time, it is necessary to reorganize the files because additions and deletions eventually distort their physical sequence to such an extent that retrieval time is adversely affected. Reorganization is accomplished by rewriting the data records in their proper sequence in another location and eliminating the original file storage area. The index files that refer to the reorganized data files must also be recreated so as to point to the new locations.

System protection

When a company commits its basic records to a data processing system, safeguards must be instituted to protect the system and the records from manipulative and from physical damage. The overall problem of authorization and a hypothetical solution are

discussed by Friedman.[2] The administrative system deals with the same problem of assuring the authenticity of system users, but incorporates an actual solution that is somewhat less flexible than the one Friedman proposes. Procedures are also built into the system for reconstructing and auditing the data base in case of either physical or manipulative damage.

authorization

Authorization is an integral part of the system design, wherein operational responsibility focuses on each individual manager whose personnel actually use the system. Assume, for example, that a manager at a given location has been delegated responsibility for accounts receivable and billing at that location. He has been given responsibility for using whatever files may have been assigned to his care on a need-to-know basis. The administrative system recognizes the manager by his employee number and security code. In the example, these identifications correspond to the manager's authorization for accounts receivable and billing—and no other applications.

Of course, the manager does not regularly perform transactions on the files; he delegates this responsibility to employees who work under his direction. Nevertheless, the manager's authorization must first be recognized by the system so that he may register his employees. Clearly, an employee cannot perform actions for which the manager is not authorized. Thus, based on the manager's registration, the employees are given security codes to perform actions delegated to them by their manager. This is the essence of the employee's need-to-know qualifications.

In actuality, neither the manager nor his employees perform actions or transactions until they have satisfactorily completed a training course for the actions in each specific transaction. In effect, the first step in authorization merely permits one to take a training course. Step by step, the student is guided through the accounts receivable and billing operations instructions on the 2260 that he will use in actual operations. Only after the student has satisfactorily completed the instruction is he permitted to perform actions in the administrative system.

When a trained user enters his employee number and security code, the system checks his authorization and training capability to perform the action he requests. If both are positive, he may begin that action.

In the case of a security error, the basic mode is for the 2260 to lock up and remain locked until a security person unlocks it. When entering his security code and employee number, the user is actually given two attempts. If he makes an error on the first attempt, the system signals him to try again. A second security error locks the machine. Attempts to perform actions for which

one is not authorized and trained similarly cause the machine to lock after a reminder is presented on the first attempt. The system is programmed to generate new security codes, which are mailed to authorized employees on a monthly basis. Although many safeguards are incorporated in the administrative system, responsibility for its success focuses on the individual manager and his employees.

data-base reconstruction and auditing

As was previously mentioned, there must be a way to reconstruct all or part of the data base in the event of its destruction. That capability is a primary task and is an automatic feature of the data management system.

Two choices were considered in designing of the reconstruction capability. The first possibility might be called "quick reconstruction" for it emphasizes rapid recovery from any data base damage. The disadvantage of this approach is that significant amounts of daily machine time are necessary to produce the rapid reconstruction capability. Since the possibility exists that it will never be necessary to rebuild the data base, all the processing done as interim steps for quick reconstruction is a wasted effort.

A second approach minimizes the day-to-day costs of providing reconstruction capability at the expense of the great amount of time to reconstruct the data base should the necessity occur. If reconstruction is unnecessary (except in rare instances), the low day-to-day cost is certainly the more economical approach.

No economical technique allows instantaneous reconstruction of the data base. Hence, "fast" and "slow" are relative terms. If the data base is destroyed, a system interruption must occur. Therefore, the amount of time required for reconstruction is not the main criterion, assuming that the amount of time can be held within acceptable limits. The advanced administrative system has implemented the second of these two choices: low day-to-day cost with higher emergency cost.

In addition to the capability to reconstruct the data base, an effective data management system should have the ability to provide an audit trail for the data base. Such an audit trail should provide the auditor with the ability to determine what information was changed, who changed it, and when this was done. The "who" can be construed to be either the program that caused the change or the individual who supplied input to the program. The design of the data management system embodies both the ability to reconstruct and to audit individual records in the data base.

The programming technique on which both the reconstruction and auditing capabilities are based is the classical accounting procedure of the journalization of all changes made to ledger

accounts with cross references (folio numbers) so that details of accounts can be reconstructed or the ledgers recreated. In the administrative system, the data base is analogous to the ledger, and all records in the data base are, therefore, ledger accounts. Any change made to any record is journalized in the data file journal and appropriate cross references made. The term "any change" must be interpreted in its broadest sense. That is, changes include not only changes to existing records (updates), but also include the creation of new records and the deletion of existing records.

Just as in the real-time operational environment, so also in reconstruction and auditing all data base activity is accomplished by the data management system. Allocated to the data management system are one or more magnetic tape drives for use by the data management system for the data file journal. When a request is made that causes the creation, updating, or deletion of any record, the data management system creates a journal record on tape. The journal record consists of the new version of the record after it has been acted upon by the user. Also included in this data file journal is the signature of the requester—program, terminal, and user identification number. Each data file journal tape is sequentially numbered from the time the system first became operational. The sequential number is incremented by one for each new data file journal tape. It is exactly analogous to the prenumbered pages in an accountant's journal.

Associated with each record in the data base is a control field for the exclusive use of data management. Included in this control field is an area to record the folio number of the last update. When a record is created, updated, or deleted, the folio field associated with that individual record is posted with the number of the data file journal tape currently being used by the data management system. Recall that the data management system writes an image of the updated record including the signature of the requester on the data file journal tape. The previous contents of the folio field associated with the record are also included in the data file journal record as the "old folio". In this way we capture in the data file journal record an exact copy of the following:

- Current version of the record
- User who caused the record to be written in the data base
- Pointer (old folio) to earlier data file journal giving the previous version of the record

Thus a complete audit from the current content of any record, back through time—update by update—may be accomplished at any time by using the folio numbers.

It is possible to reconstruct any or all files in the data base by taking all data file journals since the beginning of administrative

system operations and analyzing them to rebuild the files. This approach is impractical because of the volume of records involved. To reduce this volume to a workable size, a data file image tape is periodically created for each file in the data base. The interval of time when this image is taken varies from file to file, based upon its activity. As a file becomes disorderly as a result of activity, a reorganization of the file takes place. Concurrent with reorganization, a data file image is created. The image is an exact copy of all records in the file at the time of reorganization. The decision to reorganize is the responsibility of the data base control group. They issue the instructions to the computer operations group to perform the reorganization. The time of reorganization is recorded together with the first folio number that reflects the reorganization.

In the event of damage to any of the files, we gather together all data file journals created since a damaged file was last reorganized. The data file journals are processed by a program that copies from the data file journal tape (which contains records from all files) every record related to the file that is being reconstructed. In making the selective copy, a sequence number is added to each record for control purposes. Extracted records are sorted into descending sequence number within data base sequence. This sort puts the output in the exact sequence of the data file image taken at the last reorganization. Duplicate records (caused by updating the same record more than once since the reorganization) are together on the tape with the current record first. Duplicate records are omitted in the process because they are of no further use. The sorted records and the data file image are now processed by a program that creates a new data file that contains current records or a copy from the data file image tape.

It is not feasible to block the writing of the data file journal tapes because this would greatly increase the risk of losing several journal entries in the event of machine failure. For this reason, entries in the data file journal are written unblocked. This technique causes multiple reels of data file journals to be created each day. For protection of the data file journal as well as to conserve space, the journals are "compacted" every night. The compaction program reads the data file journals created each day, organizes them into large blocks, and creates a compacted data file journal output tape. The compacted data file journal is stored in a secure underground location.

Concluding remarks

In the advanced administrative system, we drew on the following earlier technologies that were innovative at the time we began planning the system:

- On-line data base management of airline reservation systems
- Interactive teleprocessing terminal environment of airline reservation systems
- Computer assisted instruction
- Operating system with multiprocessing capability

Analogously, the techniques pioneered or extended by the administrative system may serve as reference points of demonstrated feasibility that can lead to further advancements in information system applications. The large, varied, and highly structured data base used by the administrative system may guide others in implementing more generalized data base concepts. Administrative system use of alphanumeric terminals in a teleprocessing environment suggests the use of graphic terminals and large data base systems with greater problem solving capability. Our authorization technique, whereby thousands of users share a common data base while maintaining the security and integrity of the information, may lead to the implementation of more flexibile authorization systems.

The advanced administrative system described in this paper is, of course, a snapshot of that system in its present state of operations. Both the system and the methods are themselves subjects of continuing research and development with the intention of increasing both their scope and efficiency.

CITED REFERENCES

1. M. N. Perry and W. R. Plugge, "American Airlines SABRE electronic reservation system," *AFIPS Conference Proceedings, Western Joint Computer Conference* **19,** 593–601 (May 1961).
2. T. D. Friedman, "The authorization problem in shared files," *IBM Systems Journal* **9,** No. 4, 258–280 (1970).