RC 9659 (#42549) 10/19/82
Computer Science 10 pages

Research Report

An Integrated Office System

Christian L. Cesar
Kai-Ching Chu
Gregory A. Flurry (IBM Austin)
Brent T. Hailpern
Erik P. Jensen
Richard P. King
Henry F. Korth
Mark A. Martin
David L. Reich
Howard B. Reubenstein (M.I.T.)
Jack L. Rosenfeld
John J. Shedletsky (IBM Valhalla)
Marvin M. Theimer (Stanford)
Barry E. Willner
Peter P. Wolf (Yale)

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

IBM Research Division
San Jose · Yorktown · Zurich

RC 9659 (#42549) 10/19/82
Computer Science 10 pages

An Integrated Office System

Christian L. Cesar
Kai-Ching Chu
Gregory A. Flurry (IBM Austin)
Brent T. Hailpern
Erik P. Jensen
Richard P. King
Henry F. Korth
Mark A. Martin
David L. Reich
Howard B. Reubenstein (M.I.T.)
Jack L. Rosenfeld
John J. Shedletsky (IBM Valhalla)
Marvin M. Theimer (Stanford)
Barry E. Willner
Peter P. Wolf (Yale)

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

Successful office automation requires the integration of a variety of devices and applications in a way that is both easy to use and functionally rich. An experimental system is being built at the IBM T.J. Watson Research Center to study applications using handwriting, graphics, text, the telephone, audio, and distributed database. This paper reports on the state of this experiment, with emphasis placed on the interdependence of the various applications. The screen manager provides an integrated user interface, while the distributed database provides a uniform system-level interface for applications. Among the applications discussed are an editor that combines text, graphics and handwriting, inter-user communication facilities, and telephone control.

1. INTRODUCTION

We are designing and implementing an integrated office system on a network of personal workstations. Each workstation has a high-resolution bit-map display, a local disk, a tablet, and a speech synthesizer. The purpose of this system is to aid professionals in the preparation, storage, and retrieval of documents, to facilitate the sharing of data among users, and to provide effective user-to-user communication using diverse media.

The screen manager serves as the common user interface for all applications. It supports multiple, overlapping windows. Commands are selected by poking at "buttons" with either a tablet or keyboard. Objects, such as windows or documents, are, in general, selected by poking at them.

The distributed database system provides concurrent access to shared data and a common network view. Users may query the database using a subset of Sequel <Ch76>. The document manager provides a special interface to the database for mail and for maintenance of personal directories. Application programs use Sequel calls within their Pascal code. Among the features provided by the database are network transparency, highly-available data through replication, a general concurrency control mechanism, and a special access method designed for keyword-based retrieval.

We list below several applications included in the system.

EDITOR: The Graphic Editor treats a document as a sequence of pages containing graphical objects called constructs. Each construct may be of one of three types: text, graphics, or bit-map images. A construct can be edited using functions appropriate to its type. Constructs can also be manipulated regardless of type (moving, copying, deleting, centering, and so on).

AUDIO: Our workstations are connected to the telephone system by a machine-controllable device. This application, named Enhance-a-Phone, allows users to place calls by selecting a name or a phone number, to review a log of calls made and received, and to interface with PBX functions. In addition, Enhance-a-Phone provides a significantly improved front-end to the IBM Audio Distribution System (ADS). By using the speech synthesis function, Enhance-a-Phone serves as an advanced answering machine and call screener.

CONFERENCE: The Network Conferencing application allows pairs of users to draw simultaneously on an electronic blackboard, thus allowing text and audio communication to be augmented by pictures.

Figs. 1 and 2 show typical screen manager windows in which several applications are being used concurrently.

2. THE SCREEN MANAGER

The Screen Manager supports the creation and manipulation of overlapping rectangular presentation areas, or windows, on the display through which applications can show data to the user and receive data from the user. Applications are insulated from each other because the Screen Manager has each application see its windows as independent view spaces, irrespective of other windows owned by other applications. The applications see their interaction with the user as simple uninterrupted sequences, leaving the user free to jump from window to window as necessary.

Windows are organized in a tree structure, whose root is the whole screen. The first branches are the base visible windows. Windows may have children windows, establishing a hierarchy similar to the one in the Lisp Machine window system <We81>. The leaf windows are the only ones where data can be presented and modified, parent windows being mere containers for their children. Children of the same parent can (1) overlap one another; (2) move around independently of each other but carrying their own children, if any, along; (3) grow up to the size of their parent; and (4) shrink down to the minimal rectangle containing their own children. The state of a window includes: identity of parent, location within parent window, size, stacking relationship relative to its siblings, and the ID of the application that owns it.

Windows are created, manipulated, and destroyed by sending requests to the Screen Manager. These requests come either from applications linked internally to the Screen Manager or from applications running outside it. The former are issued through internal calls to appropriate routines, the later by sending messages to a queue that the Screen Manager polls periodically.

Input from keyboard and tablet is first examined by the Screen Manager. Depending on cursor location and state information, the screen manager either handles the input data itself (window popping, next window, etc.) or passes them to routines for the appropriate window type. These routines, in turn, either handle the data themselves or pass them to the owning application program, which decides what steps to take. The application program may simply take note of the data or issue further requests.

Window types include list, text editing, and graphic editing windows. List windows present customized arrays of buttons which the user can press to select commands to be executed, applications to be started, and so on. Text editing windows allow text files to be presented and modified and serve also as general text input and output windows. Graphic editing windows are explained in a later section.

The Screen Manager maintains two list windows of its own: a list of window manipulation commands and a list of applications. These two windows can be seen in fig. 1: commands on the left side and applications on the bottom.

Window manipulation commands include: moving and growing; picking (i.e., marking a window for further commands); popping a window to the top of the stack or pushing it to the bottom of the stack; pulling the window at the bottom to the top of the stack; hiding and "unhiding" (useful for managing screen clutter); the usual top, bottom, up, down, right, left, and left edge scrolling; deleting; refresh and reverse image. All of these can be selected and performed either with the keyboard or with the tablet.

The selection of an application in the application list normally results in the creation of one or more windows and the beginning of a new instance of the application. Some of the applications available are described in the following sections.

3. THE DISTRIBUTED DATABASE SYSTEM

Most of the applications use database functions; thus, we are constructing a fully-general distributed database system. This system is designed to take advantage of the local network for high-speed communication, and to be especially efficient for the sorts of applications we anticipate in an office environment.

We use the relational data model. For details of the relational model see a database text such as <U180>. The database is distributed among most workstations, though we do support diskless workstations. Relations may be partitioned, with partitions stored on distinct workstations. Relations, or partitions thereof, may also be replicated on several workstations. In a non-local network, read-intensive data is replicated to allow fast read access. However, in a local network, it is almost as fast to read from a remote workstation as it is to read from the local disk. Thus, we use replication as a means to obtain enhanced availability of critical data in the presence of workstation failures.

Issues of data locality, replication, and partitioning are almost fully transparent to users of the system. We say "almost," because we support a view of local autonomy which guarantees the user access to his personal data if his workstation is disconnected from the network. However, the distribution of data throughout the remainder of the network is invisible to the user. In the view of the user, his workstation is attached to a highly-reliable central system. In the event that his workstation fails, he may access replicas of his personal data using another workstation.

A primary function of the database system is to provide shared data. Since usage patterns are likely to vary, we support a flexible concurrency control scheme that supports locking at multiple granularities (ranging from all the data on a workstation down to a single tuple of a relation) and multiple lock modes tailored to specific database operations. This concurrency control scheme is discussed in detail in <Ko83>.

We chose Sequel <Ch76> as our query language. Sequel is supported in an interactive-user mode and as embedded calls within Pascal programs. This combination of Pascal and Sequel, called Passql, was motivated by the Plisql interface of System R <Ch81>. Sequel queries are passed to a procedure that returns a "cursor" on a virtual relation representing the answer to the query. Tuples are retrieved from the virtual relation one-by-one using the "fetch-cursor" call.

An example of an application using Passql to access the database is the document manager. The document manager provides both management of document libraries and electronic mail. Fig. 1 shows how the document manager window appears within the screen manager. When the "dir" button is pressed, the document manager is invoked. A large parent window appears with the child windows labelled "find and show" and "select library." The "find and show" window, as displayed in the figure, allows the selection of some documents from one library. The button labelled "some" may be toggled to read "all", which results in there being no prompt for selection, and the button labelled "one" may be toggled to read "many", thus allowing queries of multiple libraries. The "select attribute" window is used to specify a selection predicate. The result of the query appears in the large child window at the bottom of the document manager window. One may then select documents for editing or deletion. To insert, one presses the "create" button. The new information is entered in boxes that appear in the "select attribute" window.

Since keyword-based document retrieval is of prime importance to the document manager, we have designed a new access method that is optimized for queries specifying several secondary keys. The scheme, described in detail in <Ki82>, uses both extendible hashing <Fa79> and a product (logical "and") of hashes to represent sets of values.

4. THE GRAPHIC EDITOR

The Graphic Editor allows interactive creation and modification of electronic documents containing text, line drawings, and bit-map images in a unified and consistent manner. It can handle any number of documents at a time, with one edit window per document.

Documents are logically composed of pages. Only one page can be shown in an edit window at a time. The editor supports a logical page

size of up to 40 by 40 inches. Since this is larger than the screen, the window is initially positioned at the upper left-hand corner of the page. The remainder of the page can be viewed by issuing screen manager window scrolling commands.

A Graphic Editor window contains two menu areas, an output message area, and a data area (see fig. 2). The user controls the editor by poking buttons in either menu and receives status information (such as the document name) and warnings in the message area at the top of the window and/or via audio feedback. The top menu includes the commands for handling a document in terms of its pages. The bottom menu contains the commands for manipulating page contents. This manipulation takes place in the central data area where the contents of a document are presented.

The top menu includes: (1) commands for document filing -- save with or without quit and quit without save; (2) commands for leafing through a document -- moving to the first, previous, next and last pages; (3) commands for inserting a page before or after the current page, marking a range of pages, and deleting a page or range of pages; and (4) a call to a simple handwriting recognizer for use in conjunction with the tablet in entering handwritten numerals to be used as part of commands (e.g., go to page n, insert n pages, delete n pages, etc.).

The bottom menu includes commands for dealing with two-dimensional graphical objects. Such objects are called constructs. There are text constructs, various line-drawing constructs -- continuous line (for handwritten material), polyline, boxes, circles, horizontal and vertical lines -- and bit-map image constructs.

Text constructs are created and edited without the use of the menu by typing in keystrokes; but, being constructs, they are subject to any construct manipulation command provided in the menu.

Line-drawing constructs are created by first selecting the desired construct type and interactively defining the location and the extent of the construct. This, of course, takes different forms for different construct types. For example, when circle is selected, the editor interprets the first poke as the center of the circle and the second poke as a point on the circumference. While creating a construct the editor provides visual feedback about partially defined constructs: pliant boxes, lines, circles that conform to the location of the stylus between the first poke and the second.

Bit-map constructs either result from scanning printed matter with a scanner or are created by clipping a rectangular area from the current page.

Constructs in general can be directly manipulated in several ways. They can be moved, copied, deleted and aligned among themselves.

Moving, copying and deleting can be done either by enclosing the target constructs in a rectangular area or by collecting each one individually by pointing at each target construct. The first mode is desirable for manipulating clusters and the second for picking particular constructs in a cluster. Moving and copying can be done within a page, across pages, and across documents. Constructs chosen for such manipulation are highlighted before the action is performed.

5. ENHANCE-A-PHONE

Enhance-a-Phone comprises a telephone controller and a software package that interacts with the user through menu-oriented list windows and interfaces with the telephone controller and the audio system. The application provides the ability to dial by number or name, special actions for busy or no-answer calls, automatic redial, maintenance of a telephone log, automatic answering, remote telephone interactive database access (using audio response), a convenient front end to the local PBX and IBM's Audio Distribution System (ADS), and automatic delivery of audio messages to a distribution list (with responses recorded).

The application can request the telephone controller to go on-hook or off-hook on a specified connected telephone line, generate touchtone signals, connect the handset and/or an audio device (a speech synthesizer, for example) to the specified line, sound ringing signals, turn on lights, and so on. The telephone detects call progress signals (such as dialtone, ringback, and busy), touchtone signals, and information about the handset status; it reports these to the application.

The user may place a call by typing a phone number or a name. When a name is entered, the phone number associated with the name is retrieved from the database. A log is maintained for all calls, which includes phone number, timestamp, and the name of the other party, if known. During a telephone conversation, the user may take notes by typing or writing with the tablet. These notes are stored and associated with an entry in the log. The user may at any time inspect the log, read any notes, and make a call to any party in the log by positioning the cursor and poking the log entry. In a similar fashion, the user may inspect the directory and call anyone listed by poking the name of the intended person.

When a called party's line is busy or the called party does not answer, an alternate number may be called, the number may be redialed later, or a PBX automatic callback function may be invoked, if available. The most recently dialed number may be redialed at the poking of a button on the screen.

When a telephone link is established between two workstations, the called and calling workstations exchange their ID's via touchtone

signals. The name of the calling party is displayed immediately on the workstation of the called party. If the called party does not answer, the caller can leave an audio or text message.

The integration of the telephone controller and the audio system permits a great variety of functions. One such function is automatic answering. When Enhance-a-Phone is placed in automatic answering mode, any incoming call is answered with an audio message saying that the user cannot answer his phone; the caller is requested to send his ID (telephone number or special code) by means of touchtone signals. When this has been received, the system looks it up in the database and responds by (1) announcing the name of the caller over a speaker at the workstation, (2) delivering an audio message over the telephone if the user has left such a message for that caller, and (3) recording caller responses to the message. All incoming calls are recorded in a log that can be inspected as described above.

Enhance-a-Phone simplifies use of other systems that are dialed and require touchtone signals to control. Two of these are the local PBX and IBM's ADS. All PBX functions permitted are displayed in a window and any function can be invoked by poking the corresponding button. The user, therefore, does not have to remember touchtone codes. ADS functions are similarly displayed. The user can select them by poking buttons rather than using ADS's audio prompts and can perform complex functions, such as creation and maintenance of distribution lists, with ease.

An audio message can be created and delivered to a distribution list. Enhance-a-Phone calls every person on the list and plays the message to him. If a call is not completed, an attempt is made later. The response (either touchtone or audio) of the called party is saved and presented to the workstation user on demand.

6. OTHER APPLICATIONS

6.1. Network Conferencing

Users normally communicate by means of telephone or electronic mail. Network Conferencing provides the ability to communicate via a shared "electronic blackboard". When two users are in a network conference, copies of their blackboard (in a screen manager window) are kept identical. Additions to the blackboard appear simultaneously on both screens due to the high speed of the local network. This form of communication is useful to support technical discussions over the phone, to give travel directions, and (yes, we'll admit it) to play games with oth-

er users. We plan to augment the Network Conferencing application to support the full power of the Graphic Editor on the electronic blackboard.

6.2. Audio Feedback

The primary means of generating audio feedback in applications such as the Graphic Editor and Enhance-a-Phone is sending text strings to a text to speech converter (synthesizer). Its advantage is the compact storage of the messages to be spoken and the immediate use of already existing text messages and warnings produced by an application. Its disadvantage is the often poor quality of the generated utterance.

Another means is playback of voice recordings. A special hardware card captures and transforms human speech into files. These data can then be played back through the same card to reproduce the recorded speech. We use this in two ways: (1) to record long messages (coming via a telephone line or a local microphone) that are usually disposed of after being played and (2) to record isolated words or phrases that serve as building blocks for announcements -- the time of day, telephone directory information, and so on. The disadvantage of recording long messages this way is the large size of the files containing the speech data. Its advantage is the excellent speech reproduction.

In both cases the audio output can be sent either to a loudspeaker attached to the terminal or to a telephone line.

7. CONCLUSIONS

We have described the main components of our experimental office system. These components are still growing functionally. Aspects of their integration are still being worked on, such as the use of the Graphic Editor by the Network Conferencing application. But the present levels of integration have already shown the convenience afforded by having a window system that allows for running of multiple simultaneous applications and for easy and rapid switching from application to application. Important capabilities exist already: Printed matter, such as correspondence, can be scanned, added to the data base of documents, and reviewed and edited by the graphic editor. The ability of the editor to handle handwriting allows electronic documents to be signed. The telephone and ancillary aids (telephone directory, ADS, etc.) can be handled almost entirely by Enhance-a-Phone and the audio system.

The user interface for the screen manager and for most other applications is tablet oriented, but the keyboard can be used instead. The advantage of the tablet is the speed with which the cursor can be moved on the screen. Eye-hand coordination while moving the stylus to position the cursor on the screen is easily learned and the user never

has to look at the tablet, even when writing on an editor window or during Network Conferencing. But whether using the tablet or the keyboard, the selection of objects and commands by pointing to windows and poking at menus laid out on the screen is a very natural way to control many different activities going on in different windows.

Voice feedback has proven to be helpful in decreasing the amount of clutter on the screen and display activity, though its abuse can be bothersome and comprehending synthesized voice takes some practice. It is useful, for example, to inform the user of activity in an invisible or hidden window.

We believe that the technologies we have integrated will become prevalent in the office of the future. With this kind of integrated office system, more office activities will be done solely through workstations.

REFERENCES

<Ch76> Chamberlin, D.D., et al. (1976), "Sequel2: A Unified Approach to Data Definition, Manipulation, and Control," IBM Journal of Research and Development, 20:6, November 1976, pp. 560-575.

<Ch81> Chamberlin, D.D., et al. (1981), "A History and Evaluation of System R," Comm. ACM 24:10, October 1981, pp. 632-646.

<Fa79> Fagin, R., J. Nievergelt, N. Pippenger, and H.R. Strong (1979), "Extendible Hashing - A Fast Access Method for Dynamic Files," ACM Transactions on Database Systems, 4:3, September 1979, pp. 315-344.

<Ki82> King, R.P., H.F. Korth, and B.E. Willner (1982), "Design of a Document Filing and Retrieval Service," manuscript submitted for publication.

<Ko83> Korth, H.F. (1983), "Locking Primitives in a Database System," J. ACM, 30:1 (to appear, Jan 1983).

<Ul80> Ullman, J.D. (1980), **Principles of Database Systems**, Computer Science Press, Potomac, MD.

<We81> Weinreb, D., and D.A. Moon (1981), "Introduction to Using the Window System," M.I.T. Artificial Intelligence Laboratory Working Paper 210.

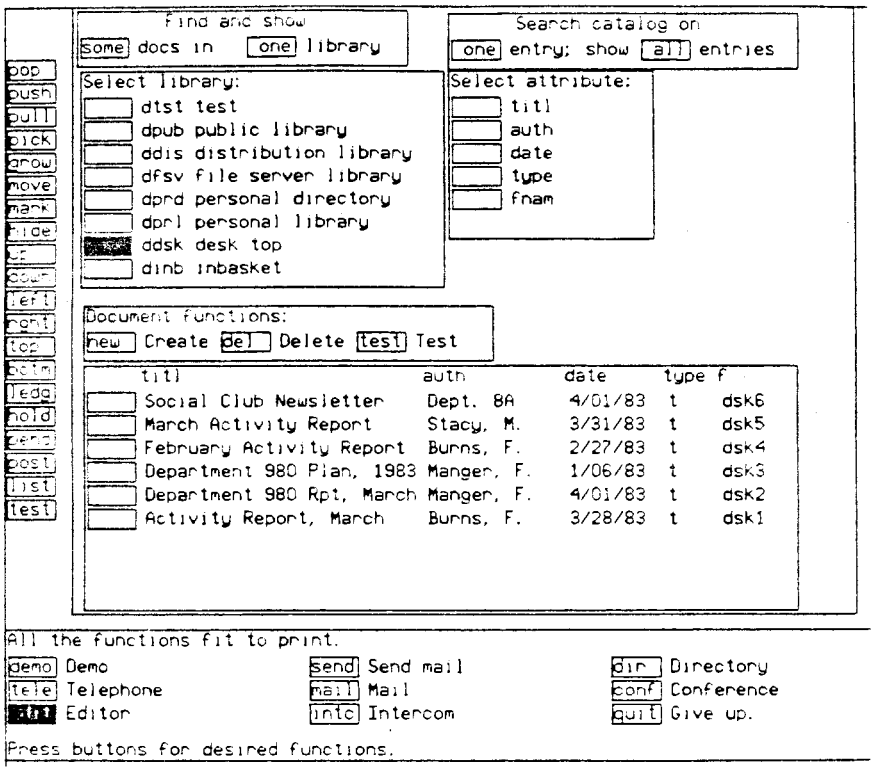

FIG. 1 Screen Manager display showing Document Manager windows

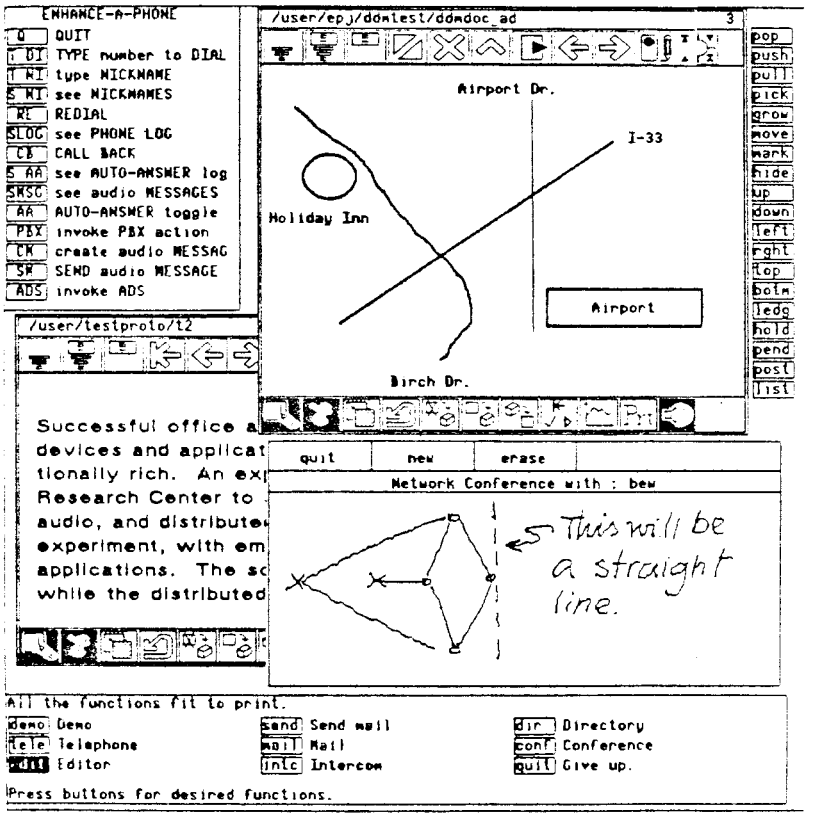

FIG. 2 Screen Manager display showing multiple applications