VITTAL

AN ELECTRONIC MESSAGE SYSTEM: WHERE DOES IT FIT?

A. Vezza and M. S. Broos
Laboratory for Computer Science
(Formerly Project MAC)
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Summary

Computer networks, such as the Advanced Research Project Agency's ARPANET[1,2,3,4], have, in the last several years, sparked the development of several different electronic writer-to-reader message systems, the harbingers of office automation systems. This paper discusses the economics and other factors relating to the cost effectiveness of such systems[5]. A message system, developed at the Laboratory for Computer Science at the Massachusetts Institute of Technology (LCS-MIT), that features extensive data base management and message processing components is described. Messages can easily be created, edited, sent, collected into sets and tagged with names of the user's choice, annotated, cross-indexed, and filed in a relational data base for easy and efficient retrieval[6]. A concerted effort has been made to provide a user interface that is understandable to those unsophisticated in computer lore.

Scenario

A worker's first task upon arriving at the office is normally to check the "in-box" for newly-arrived messages. This is accomplished not by looking into a box for pieces of paper, but by using a display console connected to a computer that, in turn, is connected to a large computer network. The worker's first action is to obtain a one-line summary of all the messages that arrived since he or she last checked the in-box. The summary contains information such as the date of the message, from whom it was sent, the first line of the subject, and the length of the message. Noticing that all but one of the newly-arrived messages are less than two pages in length, the worker defers output of the long message and instructs the computer to output the others. After each message is output, the computer pauses and allows the specification of additional instructions for actions to be taken with regard to that message.

Several of the new messages require only a simple "yes" or "no" or a short answer of no more than 10 or 12 words. The worker chooses to input and dispatch the replies to those messages immediately. He or she may attach annotations to several others and send them on to colleagues for action or informational purposes. One message is deemed to be of interest to a group of colleagues; a stored distribution list is attached to it and the computer instructed to distribute the message to the persons on the list.

Other new messages require replies of several paragraphs or a page or two in length. The first time these messages are encountered the worker may choose to shunt them to the back of the message queue. After attending to those messages requiring short replies (or no replies at all), the worker turns to the task of composing the more lengthy replies by either dictating and/or writing them in longhand. The dictation tape and/or longhand are given to a secretary who enters the replies into the computer, leaving them there for the worker to review. Later in the day the worker may decide that one of the messages entered by the secretary needs rework, so some additional instructions are dictated and the tape and message are sent back to the secretary. The others are either satisfactory, or at most require slight emendations that the worker can easily accomplish just prior to dispatching each message to its intended recipient(s).

One of the new messages requires that the worker review some background material and previous correspondence. The computer is instructed to find the pertinent material -- it retrieves three messages and the associated background material from an on-line data base: two that the worker received and one that he or she composed along with all the public comments, annotations and memoranda about the subject. After reviewing the retrieved material, a response is drafted and input as before. The newly-arrived message and the retrieved material are attached to the draft response and distributed to others within the worker's organization for comments, recommendations, and/or approval. After all of the recommendations are acted upon and/or the message is approved by the worker's superiors, the drafted response, sans attachments, is dispatched to its intended recipient(s) and a copy of the message, along with all of its attachments, is filed in the data base.

The above scenario does not take place at some distant time in the future. It illustrates daily experiences encountered by researchers at some 100 private and governmental organizations that have access to computers attached to the ARPANET. There is no dearth of message sytems on the ARPANET, as many organizations have developed their own and some hosts run two or three. Bolt, Beranek and Newman[7], the University of Southern California's Information Sciences Institute[8], and the Stanford Research Institute[9] have been active in message system development for a number of years. Currently, the MITRE Corporation, the University of California at Irvine, the Army Materiel Command, and the Naval Electronics Center are actively evaluating such electronic message systems.

The ARPANET has provided a rich environment for the development and experimentation with writer-to-reader electronic message services. Such services have provided a means for people, who are often geographically distributed, to communicate via messages sent over the network. Further, such writer-to-reader electronic message services transcend the boundary between the office environment and the delivery or postal system. Not only are messages delivered electronically, but a computational service can be provided that automates the secretary's and file clerk's functions of memorandum, letter, and document preparation, filing, indexing, retrieval, filtering, etc.

The ARPANET message system discussed in this paper was developed for a Digital Equipment Corporation PDP-10 computer known as the Dynamic Modeling System (DMS) at

LCS-MIT, and a descendant runs under a PDP-10 TENEX* operating system at the University of Southern California's Information Sciences Institute. (The DMS runs an ITS** operating system.) These two message systems go far beyond the basic services of creating, editing, printing, storing, and transmitting messages by providing a means for filing and retrieving messages by indices of originator, recipient, keyword, file folder name, date, etc. The experimental DMS Message Systems are used by some 40 staff and student researchers for message communication with other researchers. In addition, the system provides the capability for computer programs to create and receive messages. Several background processes on the DMS system, which perform tasks unrelated to the message system per se, utilize this ability, mainly for error and status reporting. One such background process even accepts a user's request for some action sent as a message to its own "in-box", and returns its results in a message sent back to that user's "in-box". In fact, two separate processes can communicate in the same manner.

Economics of Message Systems

The economics of a writer-to-reader electronic message system are governed by two factors: (1) the cost associated with creating, editing, transmitting, printing, filing, retrieving and storing of the electronically transmitted communiques; and (2) the savings, if any, that can be obtained by automating the process of preparing and handling written communications, and by reducing material costs. The current DMS Message System's cost data was not yet available as of this writing. However, cost data for an older version of the system was available and is presented here[5].

Table I indicates the cost and price of using the major services of the initial prototype of the DMS Message System for handling a typical 2000-character message. Cost was computed by amortizing the equipment over a 25-month period and adding operating expense. The tariff charged for computation by a number of service bureau organizations that offer PDP-10 time-sharing services was sampled to determine price[10]. The price range for CPU (central-processor unit) usage was small; therefore, an average is stated in the table. The price range for on-line disk storage covered an order of magnitude, therefore the range is indicated. A terabit (trillion-bit) memory with a relational data management system is not a commercial service, but the price range indicated is an estimate of what the cost and price might be if such a service were offered. The first number in each column opposite COMPOSING is the CPU charge for composing a single, typical 2000-character message. The second number is for composing additional messages, the difference being the overhead to activate the composer. The transmission cost is the estimated ARPANET transmission cost for such a message, and the price is the tariff charged by Telenet Corporation, a value-added common carrier offering an ARPANET-like service. Compared to the computational charges, the transmission charges are insignificant. The price for computation indicated in the table is in the several dollar range and therefore not very

TABLE I--CPU and Storage Costs and Prices

	CPU-SEC	COST-$	PRICE-$
Composing	14/8.9	.10/.07	1.57/1.04
Sending	8.8	.07	1.02
Receiving	10.4	.08	1.22
Filing	4.2	.04	.49
TOTAL	38/32	.29/.26	4.30/3.77
Searching & Retrieval			
1st Key	2.6	.02	.30
2nd Key	1.7	.01	.20
15 Date Keys	5.0	.04	.58
TOTAL	9.3	.07	1.09
Transmission		.0006	.012

ONLINE STORAGE

	Cost $/Month	Price $/Month
Disk	.04	.18-2.50
Terabit Memory	.002	.01- .02

CPU and storage costs and prices in dollars for a 2,000 character message. Storage overhead for indices of 25% is included. Costs are determined on a 25-month payout.

attractive. However, the measurements are from the original prototype, which was not implemented with computational efficiency in mind, but put together using, wherever possible, existing program modules. It is not unreasonable to expect that the current version provides an overall factor of two or three improvement in computational efficiency. The major improvement is in the filing, search and retrieval, sending, and receiving, with little or no improvement in composition efficiency. However, a simple experiment has shown that, by composing a messsage in the local editor of the mini-computer, central CPU composition costs could be reduced by a factor of four or more.

The figures for search and retrieval were taken from a data base consisting of some 1200 messages, indexed on 13 fields. (It is interesting to note that, although the total search utilized 17 keys, the 15 date keys were easily specified by the user, using a range notation.)

There are two other factors contributing to the apparent inflated price of sending a message. (1) The DMS Message System operates on Circa 1965 main-frame hardware even though the capital expense for currently available main-frames with 4 to 6 times the computational power is not much higher than the original cost of the DMS main-frame. (2) Commercial computer service organizations offer such a large variety of services that they are forced to mark up the basic computer cost figures by substantial amounts in order to make a modest return. One would

*The TENEX operating system was developed at Bolt, Beranek and Newman.

**The ITS operating system was developed at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology.

expect that a specialized commercial service similar to the one provided by the DMS Message System could today be offered at a price much reduced from what is indicated in Table I.

A Dartnell Corporation study[11] indicates that, using the conventional methods, the cost of creating and sending a business letter of 120 words in length (about 750 characters, which includes salutation and envelope address) was $3.79 in 1975. The following breakdown is interesting: The dictator's time accounted for 30% of the total cost, or $1.45. The secretary's dictation and letter preparation time accounted for 46% of the total cost, or $1.76, and the cost of materials, mailing, and filing accounted for 16% of the total cost, or $.59. It is hard to imagine how one could affect the cost attributable to the dictator's time except by hiring a better executive. However, it seems that it should be possible to affect costs associated with the secretary's time, the materials, mailing, and filing.

While it is true that some highly-trained individuals are able to create and edit text on-line more efficiently than someone using conventional methods, it has not been proven that current on-line text editing facilities are suitable for general office use. However, it should be noted that Word Processing is gaining acceptance in the business world. If its use by the general secretarial force favorably impacts letter preparation costs, it seems reasonable to expect that on-line message systems also can be constructed to make letter preparation even more efficient.

It should be noted that electronic message systems would almost completely eliminate the need for the services and materials listed as resulting in charges of $.59 by the Dartnell study, i.e., mailing, filing, stationery and copy costs. Using this study as a basis, electronic writer-to-reader systems would begin to become cost effective at about $.60/message. But, the study says nothing about the cost of receiving a letter, how many copies of the received letter are made for internal, secondary distribution, in how many places it is filed, etc. These costs are likely to add substantially to the $.59 figure.

Where Message Systems Fit

Where do message systems fit and when will they arrive? Are the economics such that we must wait 10 to 15 years for their arrival? Certainly, if one bases expectations of cost effectiveness on the price schedule shown in Table I then ten years appears to be about the right time frame. However, there are so many components that enter into the equation for the price charged by time-sharing companies offering general computational service that this pessimistic figure may be misleading. Taking a more optimistic attitude and using the cost figure of $.29 for composing, sending, receiving, and filing a message as a basis, a message system operating within a wholly contained environment (few or no messages need enter or leave the computerized environment) can be constructed to be quite cost effective even today. Assuming only modest economies as a result of specialized architecture, improved software efficiencies, and faster hardware, a cost decrease factor of four to ten is not an unreasonable expectation. Considering that the service is highly specialized, one might estimate that the "real cost", price minus profit markup, could be five to ten times the basic cost, or between $.13 and $.65 per 2000-character message, perhaps even less.

With an eye toward the future, ARPA and the Navy are planning a test, in an operational environment, of at least one writer-to-reader electronic message system. Even if per-message cost data for military messages existed, it is doubtful they would be available. However, most, if not all, military messages are transmitted in electronic form, indicating that a writer-to-reader electronic message service would be an excellent adjunct to the current operational message system, and the process of going from a hard-copy to electronic signal and back again is not one of insignificant cost, making it easier to achieve cost effectiveness.

In the public sector, one immediately thinks of the United States Postal Service with its 90 billion pieces of mail[12] carried in fiscal 1974. Assuming the 1968 model of mail is still appropriate[13,14], approximately 40%, or 36 billion pieces of mail, were transactions (checks, bills, statements of account, purchase orders, etc.) each containing only a few hundred characters of real information. Further, many statements of account are already generated by computer, output to paper for distribution, and the amount of the returned check keyed into the computer for use in computing the next statement. Clearly, electronic message systems may already be cost effective for handling transactions, provided the terminal and transmission system are inexpensive enough. Because there is little or no composition cost, output to a network can be accomplished as inexpensively as output to paper, and electronic input eliminates a manual operation. Clearly, from a technical point of view, a service in which all of one's statements from credit card companies, department stores, utility and fuel bills are sent directly to one's bank and accounts are settled by the individual dialing the bank's computer, receiving statement information, and keying instructions on a small, inexpensive terminal with 13 or so keys on it, is quite feasible.

With transactions amounting to the lion's share of first class mail carried by the U.S. Postal Service, is there any attraction for supplying writer-to-reader business correspondence service (which amounts to less than 10%, or 9 billion pieces of mail) with its inherent higher terminal and CPU costs? (Even less if one assumes that only businesses and government will be able to afford such services, meaning that only business-to-business, government-to-government, business-to-government, and vice versa, correspondence will be carried by such a system.) At $.13 per 2000-character message, this is a one billion dollar market. At $.40 or $.50, which is probably where it starts becoming cost effective, the market amounts to 3 or 4 billion dollars. This, however, is quite likely to be only the tip of the iceberg. The intra-organizational communications market composed of medium and large corporations may well outweigh the inter-organizational communication market by a factor of two or more, especially when the cost of duplications, etc., is taken into account. Thus, the correspondence market may well be as large as or, perhaps, eventually even larger than the transaction market, because of the additional complexity of correspondence over transactions. Will the financial institutions lead the way in office automation because of their intense interest in the transaction market?

Overview of the DMS message system

Although the two DMS message systems that operate under ITS and TENEX are not identical in terms of outward appearances and capabilities, this discussion will, for brevity's sake, assume that they are identical. Henceforth, we will

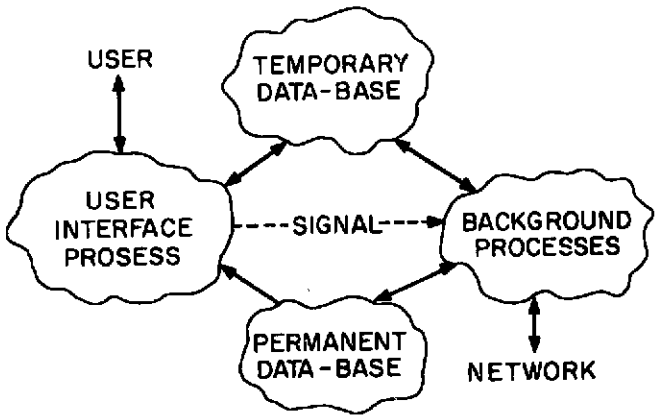

Figure 1 - A Conceptual Representation of the DMS Message System

refer to them collectively as the DMS Message System, with no attempt to point out the differences between them.

The DMS Message System was designed and implemented with the concept in mind that a message system is (or should be) data base intensive. By that we mean that an on-line data base may contain thousands or even tens or hundreds of thousands of messages. The data base must be capable of being updated frequently as new messages arrive and as users annotate existing messages or specify their own idiosyncratic filing indices. Further, the user needs the capability for finding and retrieving a message or group of messages in an easy, natural, and computationally efficient manner. For storage efficiency, parts of the data base may be shared among many users while other parts remain private to the individual. For example, all of the recipients of a message may share the text of that message, but annotations they may make to it can remain private. However, the view presented to the user is that the data base is a unified, private, personal data base, and the user neither has nor needs knowledge about which parts are shared and which are not.

Figure 1 conceptually depicts the DMS Message System. A user interacts with the user interface process, which has the ability to create other processes (notably, several different text editors), pass information to them (such as the contents of a field of a message being composed), and turn console control and data over to them. Such transitions from one process to another are carried out smoothly in response to a single command from the user, who does not need to interact with the operating system's monitor to create and pass console control to the new process. The user interface process is capable of extracting data from the permanent data base and depositing data in a temporary data base for background processes. The background processes are used to perform most of the computationally intensive tasks such as: delivering messages within and between host computers via the ARPANET, adding and deleting items in the permanent data base, addressing (the creation of a proper message header given a message and the name of a list of addressees), internal routing (automatically routing to several addressees a message originally addressed to a single addressee), message formatting, spelling correction, indexing and filing in an relational data base, etc. These are some of

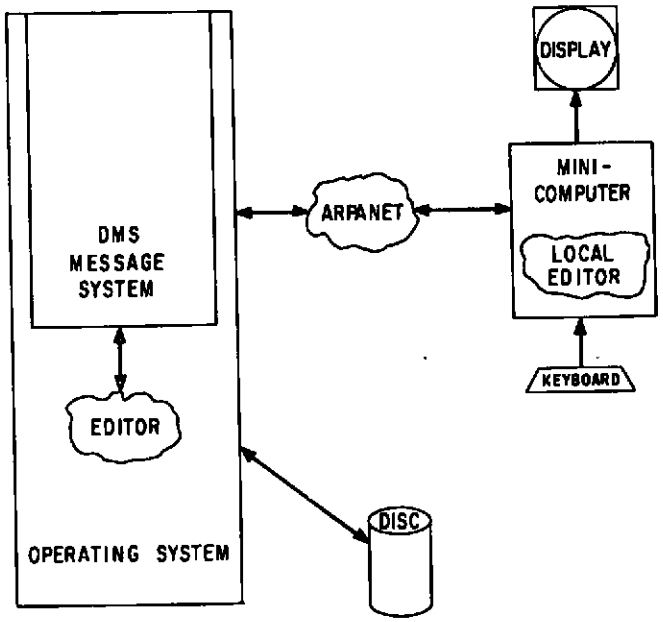

Figure 2- The DMS Message System's User Interface

the background options currently available to users of the DMS Message System. The list of possible options that could be provided is, of course, endless and limited only by one's willingness to pay for the service.

It is important that the user be presented a unified view of the data base and not be encumbered by indications of its fragmentation. The user interface process does a good job of shielding the user from any need to be aware of the existence of the temporary data base. This is accomplished by making all user commands, which conceptually reference the data base, act over the union of the two. In addition, a strong attempt has been made to present the user with an interface that is natural and easy to use. For instance, one can -- without issuing filing commands and the like -- interrupt composition, perform retrieval to obtain information, and return to composition at the point at which it was interrupted.

Figure 2 shows the basic hardware/software configuration on which the DMS Message System runs. It consists of a DEC PDP-10 computer, memory and peripherals; either the ITS or TENEX operating system; and hard-copy or display consoles. (Currently, the message system can make use of the local editing capabilities of an IMLAC PDS-1 mini-computer and display unit, and plans are in progress for interfacing it to a programmable Hewlett-Packard display unit with a special message-oriented full-screen editor developed by the Information Sciences Institute.) Access to the DMS Message System may be obtained either through the ARPANET or via a direct connection to the main frame. Character transfer rates between the main frame and the directly-connected consoles can be as high as 2000 characters/second.

Messages are composed of fields such as: subject field, date field, keyword field, text field, action-to field, carbon-copy field, etc. The DMS Message System contains groups of commands that allow insertion of information into message fields, copying of one message field into another, viewing message fields, deleting message fields, deleting messages, filing, affixing attachments, queuing messages for coordination among collaborating authors, retrieving messages from the on-line data base, printing one-line summaries, and queuing messages for transmission. The message system per se has only simple editing commands. It allows appending of information to a field and deletion of the current last character, word and line in a field. Information can be inserted from the keyboard, a file, or the on-line, relational data base. The entire contents of a field can be transferred to the editing programs TECO, XED, or to an editor in the display terminal's mini-computer. Once the contents of a field are given to an editor, the full power of that editor can be used. Upon completion of the editing process, the edited information can be returned to the field from whence it came. A display-oriented version of TECO is available which provides the user with a view of the text being edited and indicates (by a displayed cursor) the current editing position in the text. Thus, the user can see immediately the effect an editing command will have on the text. The user also receives visual feedback immediately after issuing the command, because the displayed text is updated upon completion of that command.

The User's View

The DMS Message System attempts to shield the user from having to deal with the operating system's file conventions. (Unfortunately, a user cannot be totally ignorant of the file system if he or she wishes to archive old messages onto magnetic tape.) All messages and items can be stored in the relational data base under indices and/or tags of the user's choosing. Messages stored under a common index or tag are said to belong to a set. There are a few predefined sets whose names are self describing, such as In-Box.

Figure 3 presents a view of the DMS Message System's data base as consisting of two parts: a work area, analogous to a worker's desk top, called the Cache-Area and a storage area, analogous to an office's file cabinets, called the File-Area. A message can exist in either the Cache-Area or the File-Area but not both. The Cache-Area contains all of the newly-arrived messages, messages being created, messages held for further action, and auxiliary information. Messages leave the Cache-Area when they are transmitted, thrown away, or filed in the File-Area. The Cache-Area concept actually has nothing to do with the data base implementation, but is more a conceptual construct. The Cache-Area concept does, however, provide some computational efficiency. Just as a worker might look on his or her desk for an object of interest, the worker can direct the message system to look in the Cache-Area (or even a subpart of the Cache-Area) for a message or object of interest, rather than always search the entire data base.

Each message in the data base is a member of one or more sets. Five of the pre-defined sets are mutually exclusive, i.e., a message may exist in one and only one of them at a time. These five sets can be conceptualized as physical areas in which messages reside and are called: In-Box, Held-for-Action, Discarded, In-Composition, and Filed.

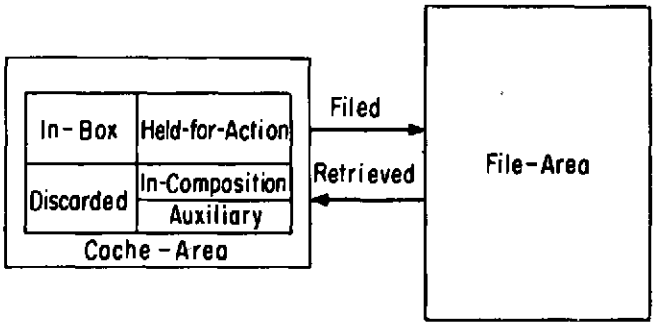

Figure 3 - Conceptual View of the on-line Data Base

The first four sets are actually subdivisions of the Cache-Area. In some sense they act like physical areas on a desk where groups of messages are kept. The Filed set is the set of all messages in the data base which are in the File-Area, i.e., those that are in the file cabinet. There is a group of predefined sets, whose members all reside in the Cache-Area, that can be thought of as tags indicating the state of a message. These sets are Summary-Not-Printed, Summary-Printed, Not-Printed, Printed, Queued-for-Sending, and Filed-During-Session. Three of the predefined sets require further explanation as to their behavior and the behavior of their member messages. When a message is discarded, it is placed in the Discarded area. It can be retrieved from the Discarded area at any time before the background process runs, at which time the Discarded area is emptied, much as a janitor empties the wastebasket. A message with a Queued-for-Sending (QFS) tag is never discarded even if it is in the Discarded area. The QFS tag is removed whenever the background process determines that the message has been successfully transmitted to all of its addressees. The QFS tag provides a good handle by which a message can be retrieved to correct an address. The Filed-During-Session tag is attached to all messages which are filed during a console session and is removed at the end of that session. The last predefined set is the Auxiliary set, which contains, not messages, but message creation templates and output format control objects.

Commands exist for moving messages from one area to another and for attaching tags to or removing tags from messages. However, the user is not really forced to use these commands. For instance, the Summary-Not-Printed tag is automatically attached to all incoming messages and automatically changed to the Summary-Printed tag if a summary of the message is printed or if the message itself is printed. Inhibiting this action is sometimes useful, as in the case of a secretary obtaining a summary or printout of the boss's messages. In this case, the tag is not changed, because the boss has not seen the summary or the messages yet, unless of course the secretary is obtaining the copy for the boss. Exactly how messages move from one area to another and how tags are changed can be individualized to a person's own idiosyncratic tastes. For instance, a person may wish to file everything automatically after printing it and

discard only those messages he or she explicitly tells the system to throw away. Another may wish to work in just the opposite manner and file only those messages for which an explicit file command is issued, discarding all others after they are printed.

In addition to the predefined sets, there are implicitly and explicitly user-defined sets, for instance, the set of all messages from Smith, the set of all messages to Jones, the set of all messages that arrived on December 10 1975, etc. Overt action on every message need not be taken to cross-file it under a group of indices. A user profile can be created which instructs the filing background process to cross-file all messages filed by user X under the indices specified in his/her profile. The user may also specify additional indices under which a message is to be cross-filed at the time of, before, or even after filing it.

The data base is so arranged that some simple questions can be answered at almost zero cost. For instance, if one is seaching for a message for which the author's name is not known precisely, a request to obtain all author names in the data base that begin with "Stein" is easily obtained. Thus, an operation analogous to scannning down a set of index tabs in a file cabinet is provided.

User Commands

The issue of whether command names should be concise and simple to enter or verbose and descriptive must be of serious concern when designing a user interface. Experienced computer users tend to prefer concise commands, with the ease of entry far outweighing their non-descriptive properties. Computer-naive users, on the other hand, find the longer, more descriptive command names more natural and easy to learn and remember. A basic difficulty in designing an interface for a computer-naive community of users is that the naive user does not necessarily stay naive. Therefore, if the target community is computer-naive, blindly choosing the verbose approach does not necessarily produce a reasonable interface. Also, the typing skills of the user community may be such as to make several-word commands awkward for them to use. The target community for electronic message systems ultimately includes almost everyone. However, the office community of clerks, secretaries, administrators and executives are the most likely non-computer type candidates to receive such systems in the near future. The characteristics of such a community are far from homogeneous, containing touch, hunt-and-peck, and non-typists. Additionally, some members of the user community may be expected to progress rapidly out of the ranks of the computer-naive, while others will not. For the above reasons, all of the commands in the DMS Message System possess long, descriptive, multi-word names and all but a few also possess short, concise abbreviations. In addition, a demand command recognition and completion feature is provided so that once enough characters to disambiguate the command, or a part thereof, has been entered, the user may, by simply typing a space, direct the message system to recognize and complete as much of the command as it can.

Table II contains a partial list of the DMS Message System commands and their abbreviations. Generally, command names consist of verb-noun/action-object pairs, i.e., the first word of all commands, except those allowing emendations to the contents of a field, is a verb signifying the action the computer is to take, with the following noun or phrase signifying the class of things to which the action is directed, e.g., Print.Message (PM), File.Message (FM), Search.Current.Set (SCS), etc.

An attempt has been made to follow a rational rule for constructing abbreviations from the command names, in order to make the transition from verbose to concise commands as easy and gradual as possible. An abbreviation is formed from the first letter of each significant word in the command name. Of course, naming collisions will occasionally prevent strict adherence to the abbreviation rule. (There is, at present, only one such collision among the 60 or so commands in the DMS Message in the instance of PTR for PrinT.Reminders. The rule says that the abbreviation should be PR, but that collides with the first two letters of all of the "Print.*" commands. Although not a collision in the strict sense, this is sufficiently annoying to the user to warrant its avoidance.) An attempt to maintain parallelism was also a definite design goal, e.g., Load.Buffer and Dump.Buffer are abbreviated as LB and DB, respectively. (Buffers are the containers used to house the contents of a field when a message is in the act of composition.)

The commands which allow one to append to a composition buffer are not of the verb-noun/action-object form. The "action", which is the same for all of the buffer-opening commands, is implied, with only the "object" needing to be specified; otherwise, all of these commands would be of the form: Append.to.<name>.Buffer or Open.<name>.Buffer. Opening or appending to a buffer does not really mean anything to a computer-naive person. On the other hand, "To", "Carbon.Copy", "Blind.Carbon.Copy", etc., are accepted terms in the office community and appear to be natural as commands directing the message system to treat the following input as the specification of that field's value. The naturalness becomes evident when you consider that a person typing a conventional memo types "FROM: <name>", "TO: <name>", "SUBJECT: <text>", etc.

Whether or not the command names, abbreviations, etc., of the user interface are acceptable to a general user community must await further testing. However, we have made some statistically non-significant observations. Non-computer people who are good touch typists (like our secretary) do not wish to be encumbered with learning and remembering abbreviations and never use the command completion feature, as they are able to type even the longest command name in two seconds or less. Conversely, people who cannot touch type will immediately search out every typing short cut they can.

To a very large degree, the DMS Message System is capable of being hand-crafted at or even after installation in order to suit each individual's or organization's idiosyncratic needs. For example, the printed message format can be stylized as to which and in what order message fields are printed, width of the display, etc. Three methods of argument prompting are available -- verbose, terse and none. All the command names, abbreviations and special characters (except the special operating-system ones) can be changed easily.

TABLE II--DMS Message System Commands

AA	Add.Annotation (to message)	PCS	Print.Cache.Summary	SF	Search.Filed (Area)
AMS	Add.Message.To.Set	PH	Print.Header (of message)	--	Send.Message (in buffers)
AN	Add.Notes (to message)	PM	Print.Message		
AR	Add.Reminder (to message)	PNM	Print.Next.Message	Composition Buffers	
AM	Answer.Message	PN	Print.News	BCC	Blind.Carbon.Copy
CAB	Clear.All.Buffers	POLS	Print.One.Line.Summary (of message)	CC	Carbon.Copy
CB	Clear.Buffer	PTR	PrinT.Reminders	FR	FRom
DC	Describe.Command	RA	Remove.Annotation (from message)	KE	KEywords
DM	Discard.Message	RMS	Remove.Message.from.Set	RE	REferences
DB	Dump.Buffers	RN	Remove.Notes (from message)	RT	Reply.To
--	exit	RR	Remove.Reminder (from message)	SP	Scratch.Pad
FM	File.Message	SN	SNDMSG	SU	SUbject
HFA	Hold.For.Action	SC	Search.Cache (Area)	TE	TExt
LB	Load.Buffers (from message)	SCS	Search.Current.Set	TO	TO
PAM	Print.and.Act.on.Message	SDB	Search.Data.Base		
PB	Print.Buffer				

The message system's commands are very flexible in the ways in which they will accept arguments. Wherever it is sensible, commands will accept multiple arguments, a range specification or a name of a set. For instance, the Print.Message command will accept "Broos, Vezza", or "21-26, 29-35", or "unread", meaning, respectively, print all of the messages in the Cache-Area from Broos or Vezza, print the messages whose Cache Id's are 21 through 26 and 29 through 35, and print all messages in the unread set.

A final comment about the user interface: It was designed with the intent that it be extremely robust and resilient. The design goal is that there should be nothing that the user can do which will cause the system to take an irreversible action that he or she will regret. For instance, aborting the act of making an entry into a composition buffer does not clear it of its contents. The exit command saves all buffers as part of its clean-up before completing the exit. The SNDMSG command (its name is historical), which provides a linked sequence of buffer commands so a user can specify a set of message fields, does not automatically clear all the buffers. Checks for insuring this robustness and resiliency must be effected in a meaningful manner. It is not meaningful to ask the user "Do you really want to discard that message?" every time a Discard.Message command is issued or "Do you want to clear all buffers?" every time a SNDMSG command is issued, because 99 times out of 100 deletion of the message is desired or the buffers are empty; asking such questions only invites the Pavlovian conditioned reflex.

The Data-Base

The overall structure of the data base is that of a single, unordered relation[6]. Each message in the data base is identified by a unique number which is used internally and never seen by the user. The indices may be viewed as sorted lists of message numbers which define subsets of the relation. (The indices are, in fact, not simple lists of message numbers, but fairly complex data structures.) The success of a retrieval request is never dependent on the existence of indices, since the user may control what message fields are indexed in his or her personal data base. Rather, those indices that do exist are used, wherever appropriate, to make retrieval more efficient. Of course, some types of retrieval requests involving non-indexed fields may, in a large data base, require large amounts of computer time to be processed. In this case, the user is notified of the potential cost and given a chance to revoke or modify the request.

The basic trade-off of an indexed (or, more typically, partially-indexed) data base is between the amount of time spent maintaining the indices and the decrease in retrieval time that such indices make possible. Fortunately, the data base used by the DMS Message System is organized in such a way that updates do not require complete re-organization of the data base. In a large data base, insertion of a single new message, along with maintenance of the associated indices, will result in the modification of only a small fraction of that data base's disk pages. The two main reasons why this is true have been mentioned before; namely, the relation of messages is not ordered and the indices are data structures, not simple ordered lists of message numbers.

The index structure is arranged in such a way that low-density keys (those representing less than 2.7% of the messages in the data base) are, in fact, represented as ordered lists of message numbers, while higher-density keys are represented as bit-masks. The figure of 2.7% is derived

from the fact that a single, 36-bit PDP-10 word in an index structure may be used to represent either a single message number or a positional bit-mask representing a range of 36 message numbers. The maximum number of operations required, Y, to merge or intersect two low-density keys is, therefore, of the order

(1) $Y = M+N$

where M and N are the lengths of the two ordered lists of message numbers. If one key is low-density and the other is high-density, then

(1a) $Y = M$

to intersect or to merge two keys, where M is again the length of the low-density key. If both keys are high-density, the number of operations required for either merging or intersecting is of the order

(1b) $Y = L/36$

where L is the total number of messages in the data base.

The above formulae do not quite predict the performance for the message system's data base because the data base is composed of structures and a single key may be of high-density over parts of its range and of low-density over others. In this case, the representation is hybrid, being a bit-mask over the high-density ranges and direct representation over the low-density ranges. Thus, the formulae provide lower and upper bounds on the number of operations necessary to intersect or merge keys.

While the actual cost of an indexed file search may be difficult to compute because of the varying densities of the keys involved, the upper bound is easily derived. Because the mapping between a key name, input by the user, and its associated index structure is made via an ordered table, a binary search may be employed, requiring a maximum of $\log_2 n$ string comparisons, where n is the number of keys in a particular index.

In the simplest case, retrieval by a single key, the upper bound on the number of operations required, X, becomes

(2) $X = C(\log_2 n)$

where C is the average number of operations required to perform a single string comparison[15]. If more than one index key is involved, the upper bound on X becomes

(3) $$X = \sum_{i=1}^{N} C(\log_2 n_i)+l_i$$

where N is the number of indexed keys involved, n_i is the number of keys in the ith key's index, and l_i is the size of the index of the ith key. (Note that l_i is always less than or equal to 2.7% of the total number of messages in the data base[15].)

We noted earlier that the figures given in Table I were taken from a data base of some 1200 messages. Assuming the worst case, where every message in that data base

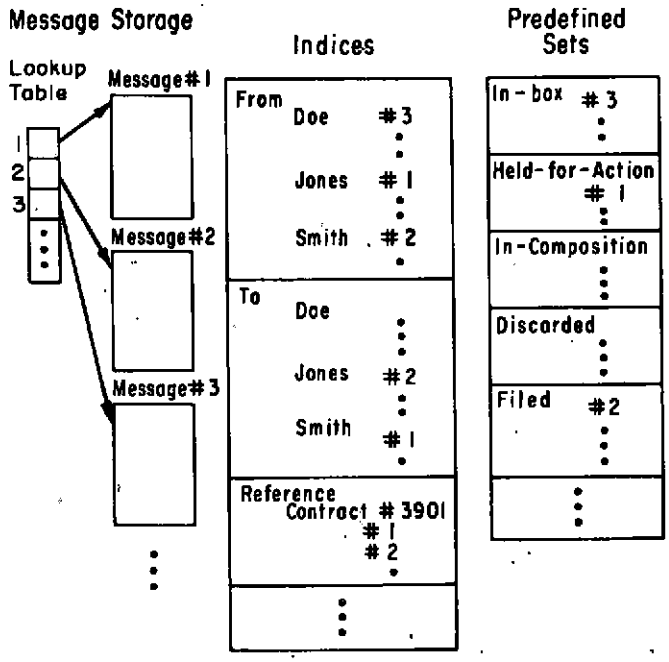

Figure 4a - Structure of on-line Data-Base

contains a unique key in a particular indexed field, the number of keys in that index also equals 1200. By Formula 1 then, the cost of retrieval on a single key becomes 10C, where C is a constant. It is interesting to note that increasing the size of the data base a hundred-fold, to 120,000 message (and keys), only increases the cost of a single-key retrieval to 17C.

Figure 4a gives a highly conceptualized view of how three particular messages, shown in Figure 4b, are represented in Smith's data base. In this example, message #2 is a reply to message #1, and message #3 has been delivered to but not yet read by Smith.

Conclusions

We have described an ARPANET message system called the DMS Message System, some of the considerations which have impacted its design, and its current economics. We have discussed such systems in a global context in order to point out what kind of driving force might be behind the realization of such systems. As microprocessors, inexpensive display units, semiconductor and bubble memory and the like herald the approach of the personal computer age, writer-to-reader electronic message systems are a natural service of the automated office, with the personal computer supplying editing and active message data base facilities, and with the inactive messages stored off on some network node. The major need will be for constructing such message systems so that they are easy for everyone to use.

Message systems such as the one described here are really the beginning of "office automation systems", because they are more than just simple creation and delivery systems. Such data base intensive systems can be naturally integrated with management information systems.

Message #1

From: Jones
To: Smith
Subject: Cost Projections?
Reference: Contract #3901
Date: December 10,1975 12:13

[Text of message #1]

Message #2

From: Smith
To: Jones
Subject: Re: Cost Projections?
Reference: Contract #3901
Date: December 10,1975 13:09

[Text of message #2]

Message #3

From: Jones
To: Project-Managers
Subject: Monthly meeting postponed
Date: December 10,1975 12:58

[Text of message #3]

Figure 4b - Examples of messages

Impediments to such services are not likely to be technological, but social, regulatory, and institutional. Clearly, mechanisms for authentication and privacy must be devised and a policy for the transition from the old world to the new must be formulated.

Finally, the models and projections designed for the volume of written communications transpiring today will be totally obsolete when writer-to-reader electronic message services become a widespread reality. Such systems will certainly create different patterns of communications, styles, and attitudes. (The authors can attest to this.) Such systems will also call for a host of new services, such as junk mail and interest filters, automated secondary distribution, spelling correction, and so forth.

Acknowledgement

The authors would like to thank E. H. Black, J. F. Haverty, M. Blank, and J. D. Sybalsky, colleagues who collaborated in the design and implementation of the DMS Message System, and S. W. Galley and S. B. Pitkin for editorial assistance.

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under Contract No.N00014-75-C-0661.

References

1. Roberts, L. G., Wessler, B. D., "Computer Network Development to Achieve Resource Sharing", AFIPS Conference Proceedings, May 1970.

2. Roberts, L.G., "ARPA Network Implications", EDUCOM, Volume 6, Number 3, pp. 4-8, NIC #12982, Fall 1972.

3. Metcalfe, R. M., Packet Communication, MAC TR-114, Laboratory for Computer Science (formerly Project MAC), Massachusetts Institute of Technology, December 1973.

4. Kahn, R. E., "Resource-Sharing Computer Communications Networks", Proceedings of IEEE, pp. 1397-1407, November 1972.

5. Vezza, A., "A Model for an Electronic Postal System", Telecommunications Policy Research Conference Proceedings, pp. 146-153, Edited by Bruce M. Owen, Published by Aspen Institute Program on Communications and Society, 360 Bryant Street, Palo Alto, California 94301, April 1975.

6. Martin, James, "Computer Data Base Organization", Prentice-Hall Series in Automatic Computation, Englewood Cliffs, New Jersey, 1975.

7. Bolt, Beranek and Newman, Inc., Hermes System Manual for Users (in preparation).

8. Tugender, Ronald and Donald R. Oestreicher, Basic Functional Capabilities for a Military Message Processing Service, Information Sciences Institute, Marina del Ray, Calif., May, 1975.

9. Engelbart, Douglas C., Coordinated Information Services Discipline- or Mission-Oriented Community, NIC #12445, Augmentaion Research Center, Stanford Research Institute, Menlo Park, Calif., December, 1972.

10. Remote Computing Directory 1974, Quantum Science Corporation, New York, 1974.

11. Dartnell Corporation, "Inflation Soars 1975 Business Letter Cost to $3.79", Analysis and Staff Report, Chicago, Illinois, 1975.

12. Annual Report of the Postmaster General, 1973-1974.

13. The Report of the President's Commission on Postal Organization, Towards Postal Excellence, U.S. Government Printing Office, June 1968.

14. The Report of the President's Commission on Postal Organization, Towards Postal Excellence, Annex, Vol. III, U.S. Government Printing Office, June 1968.

15. Vezza, A., and M. Broos, An Electronic Message System, Massachusetts Institute of Technology, Laboratory for Computer Science document SYS.16.03, March, 1976.