OSRA

OFFICE SYSTEMS RESEARCH JOURNAL

The Journal of the Office Systems Research Association
Vol. 11, No. 3, Spring 1993

The Potential of Intelligent Messages in the Automation of Office Procedures

Hossein Saiedian

University of Nebraska at Omaha
Omaha, Nebraska

Abstract

One of the focuses of artificial intelligence research has been on techniques for capturing and representing "knowledge" and "intelligence" in computer systems. In this paper, yet another application of such knowledge and intelligence representation is discussed. The idea of representing messages as intelligent objects which have certain processing capabilities that stem from their intelligence or the knowledge they have captured is explored. Since office procedures are communication-intensive and involve message passing, it would seem natural to develop an intelligent message system for automation of office procedures. A survey of prototype systems that have contributed toward the idea of intelligent message systems for use in office environments is given and issues surrounding the construction of such systems are discussed.

There has been an increasing demand for computer systems to support routine and casual office information processing activities. Under the theme of *office automation*, considerable research work has been done (Ellis & Nutt 1980; Tsichritzis, 1985; Ellis & Naffah, 1987; Noble 1991) and both office automation equipment and software packages are emerging as commercial products. Office automation in effect consists of the integration and support of various office procedures such as document preparation, electronic filing, electronic mail, administrative and decision support, etc., by computer systems. Communication systems are necessary to integrate and automate office procedures. Using the currently available office support systems and equipment, office procedures have been gradually automated, especially in routine business applications; however, they are still heavily dependent on office workers' manual handling of office procedures.

Office procedures usually require the cooperation of several office workers who may be geographically dispersed in many places. The office workers cooperate by synchronizing their actions through communication and exchange of information. Communication, thus, plays a key role in an office. Research data by Teger (1983) and Panko & Sprague (1982) show that office workers, and in particular managers and professionals, spend the majority of their time in office communication. To support cooperative office

work effectively, intelligent communication tools should be integrated in an office support system. Intelligent communication and computer-based message systems, thus, have become one of the most important issues in current office automation research.

The use of telecommunications networks and distributed systems in offices removes many constraints inherent in traditional organizational communication channels and improves many aspects of office work. They allow office information/knowledge to be passed as electronic messages, and office work to be performed by means of communication. Although the technologies developed on distributed systems are useful to support some types of office work (Mazer, 1987), they are insufficient to model some other types of office work. This is partly because communications between components in distributed systems is considered to be a very simple and structured activity (Woo & Lochovsky, 1987):

- Each component knows exactly where to get information it needs (i.e., it knows the other component that it can communicate with), and
- If a component does not know the existence of another component, they cannot communicate.

Moreover, communication in the current distributed systems is implemented in a rather ad hoc manner. For example, most systems treat communication just as transmission of some piece of text from one user to another user.

This paper discusses the development of an intelligent communication system in which messages are viewed as active objects. The goal of the research is to explore the ways to expand the capabilities of electronic communication within organizations and, thus, lead to systems that better facilitate collaborative work. This article discusses, among other things, the limitations of current computer-based messages systems for office automation, the importance of viewing office activities as patterns of messages passing, and the key role of communications systems in automating and integrating office procedures.

Viewing Office Procedures as Patterns of Message Passing

Office tasks can be classified into *structured* and *unstructured* tasks (Woo & Lochovsky, 1986). Structured tasks are of a routine nature for which some prescribed step-by-step solution exists. The phrase *office mechanization* is sometimes used to describe the process of replacing structured office tasks by corresponding automated tools (Giuliano, 1982).

Unstructured office tasks must be handled with creativity and initiative; many times they are characterized as group work involving participation of several office workers. Unstructured office tasks cannot be replaced by automated tools. *Office support systems* are built to improve the effectiveness of office workers in performing unstructured tasks. These systems, as their name implies, are built to support rather than to replace office workers.

To perform unstructured office tasks, knowledge about the nature of an office is required. As observed by Woo & Lochovsky (1986), it is not possible for each individual in a large office such as a bank to know how everything is done or to have the necessary knowledge about operations in that office to perform his/her tasks. Office knowledge/information is distributed among office workers (or other entities). Furthermore, this knowledge is constantly and dynamically changing. As a result, office workers need to cooperate to accomplish their activities. Cooperation is achieved through exchange of information and knowledge. Exchange of information most likely takes place within a communication system. The office knowledge is, thus, disseminated by means of a communication system among office entities. A message is normally referred to as a unit of communication. Messages are passed among office entities for a variety of reasons, for example:

- Office entities perform their activities concurrently. Messages may provide a means for office workers to synchronize their activities if there is a need for synchronization.
- Office entities perform their tasks using a set of input from some other office entities and they may produce a set of output which could then be passed to other office entities. Messages may serve as means to represent such input or output sets.

Messages may be interrelated. For example, to process a message, an office worker may have to wait for the arrival of another message. The delay in arrival of a given message may cause other messages to wait. These message dependencies are inherited directly from the nature

of unstructured tasks. Furthermore, the response of an office worker to a given message cannot be predicated in advance. This implies that further message passing has to take place among other office entities to achieve some objectives or to solve a problem. Consider, for example, the problem of locating information. Woo & Lochovsky (1986) observe that as the size of the organization increases, this can become fairly complex. Because information flows among office entities, it cannot be guaranteed that some information can be found once all locations have been visited. To locate the information, an *object* is needed that takes the responsibility of traveling, communicating, and bringing back the required information. This requires an active (message) object with intelligence to make decisions about which location to visit next.

Messages are delivered to office entities by means of communication systems also known as *computer-based messages systems* (CBMSs). The successful completion of office tasks relies on the effectiveness of the underlying communication system. Naturally, as more responsibilities are placed on the communication system, more efforts of office workers can be saved with more time to concentrate on creative work, and productivity should increase. Currently, communication systems in offices are passive in that the knowledge required for communication resides with the users. The communication system is only responsible to deliver messages from one user to another. To solve office problems by means of message passing, however, requires more powerful communication systems. These concepts are discussed in the next section.

Computer-Based Message Systems (CBMSs)

Computer-Based Message Systems (CBMSs), also known as electronic message or mail systems, allow two or more individuals (or entities) to communicate electronically. In these systems, computers serve both to mediate the actual transmission of messages between entities involved as well as providing the users with facilities to create, read, route, or destroy messages. The advantages of CBMSs can be summarized as follows (Bruder, Moy, Mueller & Danielson, 1981):

- Allow asynchronous communication in which the expected recipient(s) need not be ready at the time of message transmission.
- Users are released from geographic restriction.
- Sender's and receiver's times are optimally used.
- Reliable and speedy transport of messages is provided.

The users of a CBMS exchange messages for essentially two reasons: to deliver information and to collect information. In the former, the messages are complete in themselves while in the latter, messages that are intended for collecting information are followed by some action by the recipient(s) sometimes after the message has been received. A CBMS is essentially a distributed system. A distributed system is built on top of a computer network. A computer network is a collection of computers that are interconnected and can communicate with each other.

Computer networks can be organized in many different ways. A well-known model of computer network organization is the International Organization for Standardization (ISO) Reference Model for Open Systems Interconnection (OSI) (Black, 1991). The ISO OSI model has seven layers where each layer consists of a process or a group of processes. Conceptually, each layer communicates with the corresponding layer on another machine. The ISO OSI layers are called application, presentation, session, transport, network, data link, and physical layers. The application layer is at the highest level while the physical layer is at the lowest and is responsible for actual transmission of data bits. A CBMS is an example of a program at the application layer.

A CBMS provides features that allows a user, the sender, to prepare a message and send it to one or more other users, the receivers. Once the sender gives the message to the CBMS, the CBMS would have the responsibility for maintaining the integrity of the message and delivering it to the expected recipients. Once the message has been delivered, the recipients will be responsible for processing it. As observed by Wong (1990), an aspect of CBMS is the convenience of using a third party carrier between the sender and the receiver. Without the CBMS, senders must each individually gain access to receivers' mail space (which is normally a private area) to deliver a message. By using a CBMS, senders can, however, rely on the services of the CBMS to gain access to and place the message in the receiver's mail

area. Mackay (1988) provides a taxonomy of work functions accomplished in an office environment by a CBMS. These are summarized by Wong (1990):

- *Information Management*: Office work involves processing and exchange of large amounts of information. A CBMS has the potential to improve an office worker's abilities to receive, process and maintain large amounts of information and can help the office worker in information collection, compilation, filing and retrieval. Useful information from a wide variety of sources can come in as messages. A CBMS allows messages to be organized according to the factors such as the time of arrival and the source of the information. The CBMS may archive some messages for future use. Thus a CBMS message may serve both as a source and as a repository of information.
- *Time Management*: Office workers perform a large number of tasks each day. These tasks may have been assigned electronically or they may be electronically-based tasks. Since many of these tasks may be assigned electronically via messages, the office worker must be able to make some assemblage as to which tasks are more important. They can use a CBMS as a time management tool in identifying and prioritizing task-related messages. A CBMS may effectively and efficiently help office workers to identify and sequence important messages. This is necessary to minimize the amount of time spent on performing a single task and to ensure that a critical task is processed on time. Furthermore, it is by far more efficient (both time wise and cost wise) to send a message or to reply to a message using a CBMS instead of using traditional methods such as postal mail, telephone calls, or hand delivery to a person on the next floor.
- *Task Management*: An office task normally requires the cooperation of two or more office workers. The office workers may perform their tasks using a set of input from some other office entities and they may produce a set of output which could then be passed to other office workers. Messages may serve as means to represent such input or output sets.

Since an office is a distributed work environment, it is more convenient to use a message to carry a request or some input data to an office worker. An office worker may send a message to request an action from another office worker. Thus a message can be viewed as a unit of work and, thus, task and subtask management can be done with the CBMS more efficiently.

The CBMSs have proven to be invaluable communication tools to computer users in various organizations. They have created new forums for exchanging information between individuals who have never met. Furthermore, message systems have become an important area of research in office automation. Since offices are communication intensive environments, computer-based message systems are an integral part of any office information system. A discussion of exploration of communication techniques as it is relevant to office information systems is given by Woo & Lochovsky (1986) and Ellis & Naffah (1987).

The Problem

Traditional computer-based message systems are passive systems in which users have to initiate all actions. In particular, messages are passive entities which consist strictly of data with no processing power. In a typical CBMS, users are required to know all the expected recipients of a message and then have to explicitly specify the message routing path and other message management functions. The message route would be point-to-point, i.e., the sender of the message specifies the most intermediate destination from which the next user must specify the next action, if any, and so forth. If a distributed list of recipients is specified by the sender, a copy of the message is sent to every member of the list. However, every receiver has to initiate further actions if needed. Even though electronic message systems are more convenient and offer greater flexibility and power than regular mail systems, these systems should comprise more than just elementary editing and transport functions and that their real capabilities should be fully exploited. What is being proposed is that, unlike the traditional mail systems in which messages are treated as passive entities of data with no processing power, a facility should be incorporated into message systems so

area. Mackay (1988) provides a taxonomy of work functions accomplished in an office environment by a CBMS. These are summarized by Wong (1990):

- *Information Management*: Office work involves processing and exchange of large amounts of information. A CBMS has the potential to improve an office worker's abilities to receive, process and maintain large amounts of information and can help the office worker in information collection, compilation, filing and retrieval. Useful information from a wide variety of sources can come in as messages. A CBMS allows messages to be organized according to the factors such as the time of arrival and the source of the information. The CBMS may archive some messages for future use. Thus a CBMS message may serve both as a source and as a repository of information.
- *Time Management*: Office workers perform a large number of tasks each day. These tasks may have been assigned electronically or they may be electronically-based tasks. Since many of these tasks may be assigned electronically via messages, the office worker must be able to make some assemblage as to which tasks are more important. They can use a CBMS as a time management tool in identifying and prioritizing task-related messages. A CBMS may effectively and efficiently help office workers to identify and sequence important messages. This is necessary to minimize the amount of time spent on performing a single task and to ensure that a critical task is processed on time. Furthermore, it is by far more efficient (both time wise and cost wise) to send a message or to reply to a message using a CBMS instead of using traditional methods such as postal mail, telephone calls, or hand delivery to a person on the next floor.
- *Task Management*: An office task normally requires the cooperation of two or more office workers. The office workers may perform their tasks using a set of input from some other office entities and they may produce a set of output which could then be passed to other office workers. Messages may serve as means to represent such input or output sets.

Since an office is a distributed work environment, it is more convenient to use a message to carry a request or some input data to an office worker. An office worker may send a message to request an action from another office worker. Thus a message can be viewed as a unit of work and, thus, task and subtask management can be done with the CBMS more efficiently.

The CBMSs have proven to be invaluable communication tools to computer users in various organizations. They have created new forums for exchanging information between individuals who have never met. Furthermore, message systems have become an important area of research in office automation. Since offices are communication intensive environments, computer-based message systems are an integral part of any office information system. A discussion of exploration of communication techniques as it is relevant to office information systems is given by Woo & Lochovsky (1986) and Ellis & Naffah (1987).

The Problem

Traditional computer-based message systems are passive systems in which users have to initiate all actions. In particular, messages are passive entities which consist strictly of data with no processing power. In a typical CBMS, users are required to know all the expected recipients of a message and then have to explicitly specify the message routing path and other message management functions. The message route would be point-to-point, i.e., the sender of the message specifies the most intermediate destination from which the next user must specify the next action, if any, and so forth. If a distributed list of recipients is specified by the sender, a copy of the message is sent to every member of the list. However, every receiver has to initiate further actions if needed. Even though electronic message systems are more convenient and offer greater flexibility and power than regular mail systems, these systems should comprise more than just elementary editing and transport functions and that their real capabilities should be fully exploited. What is being proposed is that, unlike the traditional mail systems in which messages are treated as passive entities of data with no processing power, a facility should be incorporated into message systems so

that messages can be envisaged as active objects with programmable *intelligence*. In other words, a framework for message systems is suggested in which messages are *active* and *intelligent* which, among other things, can

- collect responses from their recipient,
- make certain routing decisions predicated in the programmable intelligence,
- modify themselves at intermediate routes before continuing their route,
- have a "mission" but can learn from their environment,
- return to originator if necessary,
- augment the message sender's knowledge about the recipient,
- actively seek information when necessary,
- autonomously check for exception conditions, and
- provide additional security.

Thus, it can be argued that active message objects are a natural mechanism for building an advanced message management system. The objective of this paper, therefore, is to capture a notion of an intelligent message object that can achieve the above goals.

The basic ideas of the proposed model are in some ways very simple, and in some ways very complex. Many of the features of the proposed model may not be very difficult to prototype individually, but perhaps they will be hard to implement as a collection. The following sections provide some background on the above topics, relate author research to existing work in the literature, elaborate on potential advantages and significance of viewing messages as active objects, and discuss the theoretical issues that need to be formally addressed. In particular, speculation is provided on the complexities of developing a formal framework which is powerful enough to capture semantic properties of local and global behavior of active messages divorced from any particular implementation.

Background and Relation to Other Work

This study of "message objects" is related to the research concepts in the area of object-orientation. The concepts of *objects* and *active* messages are not new. These and related topics are briefly reviewed in the following sections.

Object-Orientation

Object-orientation is a trend toward software development that has emerged in recent years. In this framework, a complex system is viewed as a collection of objects. Each object is considered to be an autonomous and self-contained entity that represents a physical or abstract entity. One main goal of object-oriented framework is to maintain a direct correspondence between the real-world entities and their representations in computer systems being developed so that objects do not lose their integrity and identity. The object-oriented approach provides advances towards software engineering concepts such as abstraction, modularity, reusability, rapid prototyping, and so forth. A recent issue of the *IEEE Computer* (October, 1992) provides several excellent technical articles on object orientation. Other good references include two books, one edited by Kim & Lochovsky (1989) and one written by de Champeaux, Lea & Faure (1993).

Active Objects

The earliest theoretical work to characterize the behavior of active objects was the *actor model*. In this model, "actors" and "events" are the fundamental concepts. Actors are referred to as computational agents while events mark the arrival of messages at an actor. Actors interact with each other through one actor sending a messenger (or a message) to another actor. The key point in the actor model is that messengers were themselves actors or active objects. The objective is to present a model in which everything was an object. However, this approach, taken to an extreme, would lead to an infinite regression since the only way two actors would interact would be by sending messengers. But if a message itself is an actor, then one must send it a message, and so on. Agha (1986) provided a formal definition of the actor model in which messages were no longer treated as actors but as abstract communications. Agha argued that if messages were viewed as actors, formal definition of semantics of the actor model would become too complicated.

Active Messages

Research-To-Development-Tool (R2D2). The earliest work in this area belongs to Vittal (1980). Vittal described an experimental system called the Research-to-Development-Tool (R2D2) in

which messages are capable of performing certain actions on their own. In Vittal's system, a message is treated as a *messenger* by allowing a simple routing specification to be stated in a distribution list in that message. Each active message routes itself according to the *Circulate-Next* instruction in the list. Messages can also tailor their interactions with a user, depending on the responses they receive from the user.

Conceptually, Vittal considers an active message to be a single, self-modifying entity and allows a simple routing specification as part of an *ad hoc* serial distribution list. This capability is limited, allowing a simple routing specification and no decision criteria. Furthermore, Vittal does not discuss issues related to dynamic routing and problems associated with multiple copies. The development of R2D2, however, suggested that the capability of a message system can be increased by allowing a message to have a continuous motion, moving from receiver to receiver.

Imail. In the *Imail* system of John Hogg (1985), a message is allowed to interact with its recipient, and on the basis of the responses that it collects, the message decides whether to route to further recipients or to terminate. Imail messages are programs and Imail commands are embedded into the messages called *Imessages*. Each Imessage consists of a list of questions in its *script*. The Imessage script is translated into UNIX C shell programs. It is the C shell programs which are actually executed at the recipient's site. While the program is being executed, *interaction* takes place between the recipient and the Imessage. During the interaction, the Imessage collects information which can be used later. The user is responsible for message management, however. For example, the Imessage is not able to automatically identify other potential recipients that are to be added to a current recipient list as a result of dialogue with the user. Users have to decide which receivers have to be added to the recipient list.

Since an Imessage is a program, a user receiving the message has to run the Imessage program to initiate its execution. Once the Imessage has been executed (or asked to run), it begins its interaction. Furthermore, the users are required to supply the information asked by the Imessage program for the Imessage to continue its operation. These imply that user's involvement is crucial, otherwise the system cannot guarantee that information delivery and information collection can be done in a timely manner. Thus, the Imail system also places the responsibility of message management on the user although its Imessages can make certain routing decisions.

A similar system has been proposed by Wong (1990) in which messages are called *Amessages*. In the *Amessage* model, a static rule-based framework is incorporated into messages that allows a message to determine the routing which it has to undergo. The time-dependent constraints in this model are specified and verified via a variation of temporal logic language.

Message Management Systems. Message management systems typically include a database to organize the messages and a set of message management rules. The message management rules represent the receiver's desire of what to do with an incoming message. The message management rules can be applied to an incoming message and appropriate action can be performed automatically. A good example of such a system is the MMS system (Mazer & Lochovsky, 1984). The MMS system is conceptually an integration between a database system and an electronic mail system and supports logical routing specifications where the routing of a form or a document through the system is based on conditional rather than predefined routing. The main feature of this system is the ability to either specify routes for a message or to use embedded knowledge of the state of the message contents and system conditions to make decisions on the correct route.

The logical routing model of MMS does not view messages as objects with capabilities mentioned earlier, but allows a "logical routing" in which the system assumes responsibility for evaluating the current message instance state to yield the next destination, thus freeing the user from the need to direct each instance of a message. However, the system is designed using a single message to solve the problem at hand and this MMS lacks facilities to handle interacting messages.

Several other message management systems have been developed. These include a Knowledge-based Message Management System (Chang & Leung, 1987) and the Information Lens (Malone, Grant & Turbak, 1986). The system developed by Chang & Leung (1987) is an extension of the MMS

that includes a knowledge-base and a filtering system. The messages are grouped into immediate, general, and junk messages. This grouping is based on things such as message length, traffic constraints, etc. Before a message leaves the sender's station, the filtering system deletes junk messages. Immediate messages are delivered in full to the destination.

General messages are truncated appropriately and then transmitted as alert messages. The knowledge base contains the alert rules. The alerting rules have the form IF CONDITION THEN ACTION. If the condition is satisfied, the corresponding alert rule will be triggered and the action will be executed. The actions include filing, mailing, and database retrieval. The main objective of this system is to control junk mail from flooding office workers.

The Information Lens (Malone, et al., 1986) is a prototype mail system which is similar to the above system, i.e., it helps office users in filtering and organizing electronic mail messages. Users can develop their own sets of IF-THEN rules, and the Lens system processes incoming messages according to these rules. The rules can perform operations such as moving a message to an appropriate folder or to identify characteristics of "interesting messages."

Discussion

Some other models provide an automatic routing facility (Shu, 1982; Tsichritzis, Rabitti, Gibbs, Nierstrasz & Hogg, 1982). These models allow implicit destination specification and, thus, free the users from explicitly specifying each destination for each message instance sent. Other models, e.g., Chang & Chang (1982), use a database alerting technique for office activities management and routing destination specifications are formed as actions in alerts.

Most of these models have typically taken a narrow view of an intelligent message management system in which message routing is typically single hop, i.e., from the sender to the most immediate destination and each user must explicitly interact with each message as it flows around the system or else the routing decisions are static. Furthermore, none of the above models have fully investigated the concept of *messages as objects* in which messages would have similar capability and power of objects in object-based systems.

Significance of Viewing Messages as Objects

A message object is essentially an abstract data type with instructions to route and process a message with user interactions embedded in the message object's *script* (i.e., the executable part of a message). As a result, a message object can take a role analogous to a processor executing instructions of a program and, thus, it can directly execute the actions which are needed to perform the task assigned to it by its originator. For example, when a message object arrives at a given station, it can execute the code associated with that recipient station. The actions may also have time-dependent constraints; for example, if there is a specific time difference between the creation time of the message and the current time, the message object would terminate its mission and archive itself in the originator's station. Also, since a message object may have routing information in itself, it can deliver itself to a next station if the current station doesn't interact with it after some period of time. This approach would also provide a better user interface since the users are released from the burden of providing details for routing procedures. Since messages are objects, they will have an "identity." Thus, one can communicate with them. This implies two added features. First, if a user has sent a message but does not wish the message to be seen by the intended recipient, the user can send another message to the first message asking it to destroy itself or to return (provided the receiver has not received the message yet). Second, two message objects may communicate with each other. This feature allows cooperative work among messages that are part of the same activity.

To provide a more flexible user interface, a message object may have a number of state variables (fields) similar to the ones proposed by Gehani (1982) and Tsichritzis et al. (1982). For example,

- *personalized* fields which are automatically filled from the sender's profiles and system variables, (e.g., name, station number, date),
- *required* fields which must be entered by the sender when creating a message object,
- *unchangeable* fields which consist of optional data but cannot be changed once entered,
- *unrestricted* fields which

include optional data and can be modified at any time and by any user,

- *virtual* fields which are automatically computed by the system according to some computations rules or preconditions specified in the message object's script,
- *ordered* fields which can only be filled by the users after some other fields have been filled,
- *key* field which is for system identification purposes and is generated by the system when a message object is created and may not be modified by the users,
- *lock* fields, which, if filled, result in certain other fields being protected from modification,
- *invisible* fields which are invisible to users who do not have access rights to read/modify a message body, etc.

The need for some or all of the above fields can easily be justified. For example, the *invisible* fields essentially enforce the *access rights* specifying users authorized to access, create, destroy, copy, or modify certain message objects. The access rights might be at field level (e.g, certain users may not modify a specific field on a message object) or at the entire message class level (e.g., certain users may not create certain message objects). Current research focuses on the problems discussed earlier with the belief that new results are obtainable in each of the areas. The most difficult problem will be the development of a formal model of computation which considers messages as objects. The theory developed would enable messages to be considered as "first class" objects with respect to computations and, thereby, endow them with the power that messages in human communication possess. Thus, messages can be assigned to other variables, can be passed as parameters for a computation, and can be the result of a computation. These capabilities enable the notion of higher order messages.

Practically, the significant aspect of the research would be the development of a secure and active message system, which could form the nucleus around which many of the office automation products would be developed. This research could spawn a significant number of software products for the office of tomorrow. The proposed message system takes security one step higher in the sense that not only does it prevent unauthorized access, but also hides the very existence of the message. This can also lead to some results in the area of public key cryptography where the problem is to exchange some vital information over a public access network.

A Computational Model

The message system consists of one or more sites. Each site has computational facilities and, in addition, facilities to communicate with the other sites. One of the computational facilities at each site is the mail manager. The mail manager is the interface between the users at the same site or at other sites. The mail manager provides the user at a site facilities to send and receive mail. Each mail message is a 2-tuple consisting of both the data sent by the sender and the mechanisms (or operations) to interpret the data. In other words, each mail can be viewed as an instance of an abstract data type containing data and operators for this data. To send a mail message, the user sends the mail data and mail operators appropriately encapsulated to the mail manager with the name(s) of the intended recipient(s). The mail manager uses the communication facilities to transport this mail, if required. At the receiving site, the mail manager receives a mail message from its communication facility. The mail manager then informs the recipient of the arrival of a mail message. When the recipient desires to read mail, this request to the mail manager is passed to the received mail by the mail manager. The received mail then interprets the recipient request as an operator and processes this request using the mechanism supplied by the sender on the data supplied by the sender. The results of the processing are then sent to the recipient (either directly or through the mail manager). Since reading of the mail initiates a computation (coded as part of the mail), further computations may be generated with or without the knowledge of the recipient. For example, the sender may code as part of the read command an action which has the effect of sending an acknowledgment of mail having been read without the recipient knowing about it or it may explicitly solicit a response from the recipient and send it either to the sender or to some other recipient. As an extreme action, it may even destroy itself (the mail) to preserve secrecy. This model fits equally well in either a sequential

processing environment or a concurrent processing environment. In the case of a sequential processing environment the mail is reduced to a procedure executing at a remote site on local data (local, since the data is sent along with the procedure) under the control of the mail manager. Although similar to remote procedure calls (RPC) it is not the same as RPC. In a sequential processing environment, the mail manager has to act as the intermediary between the recipient and the sender's mail. In a concurrent processing environment, the mail message can set itself up as a separate process on arrival at the remote site and start communicating with the mail manager. Similarly, each user's interaction with the mail system can set up a new process. In this case, the mail manager only acts as a liaison between the two processes (the user process and the process corresponding to the received mail) and its job is done once the two processes are informed about each other's existence.

This proposed model differs from the traditional mail system model in two ways. Traditional mail systems consider mail to be a sequence of characters, whereas, the model considers a mail item to be a computational entity. Secondly, in the traditional mail systems, the sender's mail becomes recipient property (or data) after it is received by the recipient. In the model proposed here, the mail message retains its identity on arrival at the receiver's location and remains a distinct computational entity. (One can, however, simulate the behavior of the traditional mail system by coding this action as part of the mail.)

There are quite a few questions to be answered with respect to the model. The foremost question concerns a formal model for the message system. One approach is to consider the mail as an object and then extend the actor model (Agha, 1986). The model described by Agha does not include messages as objects.

The other questions relate to implementations of the message system. In the case where the mail system is distributed over a geographic area, the underlying computational environment is necessarily loosely coupled. In this case, the computational resources at the different sites may be homogeneous or non-homogeneous. If the computational resources at the different sites are homogeneous, then it is possible to send the operations part of the mail in the executable format. However, this raises a problem with respect to the communication of the mail message. The problem is related to the fact that part of the mail is binary (the operations) and part of the mail is text (the data). In a heterogeneous system one has to decide on a methodology to allow operations to execute at a remote site running under a different computational environment. One approach consists of defining a set of primitives which are supported by the mail managers at all the sites and then coding the operations via these primitives (similar to the file transfer protocol primitives). This raises further issues, namely, the constitution of an adequate set of primitives and mechanisms to write user operations using these primitives. Depending on the size of the set of primitives, one could either enhance the mail manager to provide a user interface which would convert user actions into primitives or, in the case of a large set of primitives, one may need to think in terms of defining a new programming language catering to mail-oriented applications.

Conclusions

The study of "message objects" can be viewed as analogous to the study of object-oriented programming languages. The intention, however, is to provide an intelligent way of representing messages and communications for office automation purposes. The notion of "active" and "intelligent" message objects was proposed. Messages are active and intelligent because of the capabilities with which the messages are endowed and because of the way the messages interact with the users. This approach opens new possibilities for implementing office procedures and suggests that delegation of responsibility can be performed in more distributed and natural ways.

It is necessary to understand the properties of a formal model for representing messages as objects before an advanced communication management system can be built with all the features discussed in this paper. Message objects can be used as an intelligent way or as a framework for an advanced communication system. Issues such as formal definition of semantics, completeness of the system, etc., need to be fully investigated so confidence in such a communication system will increase.

A main challenge is the formal semantics definition of the intelligent objects in the proposed

CBMS. The main goal of a semantic model in this context would be to capture natural structure, and describe the properties and plausible restrictions in a communication system that are physically realizable. A semantic model would formally and mathematically document the structure of the message system at varying levels of detail, would aid users in interpretation of the system, and would allow proposed advance message system to be specified, developed, and verified in a systematic rather than *ad hoc* manner.

A number of formal models, most notably the actor formalism as defined by Agha (1986), are being currently investigated as the semantic model for the proposed message management system. The actor model not only captures the abstract power of the object-orientation but provides as well a mathematically precise abstract machine for analysis of asynchronous and concurrent computations based on message passing. The actor model has been used for modelling office entities (Saiedian, 1991) and for the specification of office tasks (Saiedian, Farhat & Zand, 1992).

Acknowledgment. The author wishes to thank the editors and anonymous referees for helpful comments and corrections. The author also wishes to acknowledge A. Ravichandran for his contributed ideas.

References

Agha, G. (1986), *Actors: A Model of Concurrent Computation in Distributed Systems*, MIT Press.

Black, U. (1991). OSI: *A Model for Computer Communication Standards*, Englewood Cliffs, NJ: Prentice-Hall.

Bruder, J., Moy, M., Mueller, A. & Danielson, R. (1981). User experience and evolving design in a local electronic mail system, in R. Uhlig, ed., *Computer Message Systems*, North Holland Pub., pp. 69-78.

Chang, J. & Chang, S. (1982). Database alerting technique for office activities management, *IEEE Transactions Communications*, COM-30(1), 74-81.

Chang, S. & Leung, L. (1987). A knowledge-based message management system, *ACM Trans. Office Information Systems, 5*(3), 213-236.

de Champeaux, D., Lea, D. & Faure, P., eds (1993). *Object-Oriented System Development*, Reading, MA: Addison-Wesley.

Ellis, C. & Naffah, N. (1987). *Design of Office Information Systems*, New York: Springer-Verlag Publishing Co.

Ellis, C. & Nutt, G. (1980). Office information systems and computer science, *ACM Computing Surveys*, 12(1), 27-60.

Gehani, N. (1982). The potential of forms in office automation, *IEEE Transactions on Communications*, COM-30(1), 120-125.

Giuliano, V. (1982). The mechanization of office work, *Scientific American*, pp. 149-165.

Hogg, J. (1985). Intelligent message systems, in D. Tsichritzis, ed., *Office Automation*, New York: Springer-Verlag.

Kim, W. & Lochovsky, F. H., eds. (1989). *Object-Oriented Concepts, Databases, and Applications*, Reading, MA: Addison-Wesley.

Mackay, W. (1988). Diversity in the use of electronic mail: A preliminary inquiry, *ACM Transactions on Office Information Systems, 6*(4), 380-397.

Malone, T., Grant, K. & Turbak, F. (1986). The information lens: An intelligent system for information sharing in organizations, in *Proceedings of the CHI 86 Conference on Human Factors in Computing Systems*, pp. 1-8.

Mazer, M. (1987). Exploring the use of distributed problem solving in office support systems, in *Proceedings of the IEEE Symposium on Office Automation*, pp. 217-225.

Mazer, M. & Lochovsky, F. (1984). Logical routing specification in office information systems, *ACM Transactions on Office Information Systems, 2*(4), 303-330.

Noble, F. (1991). Seven ways to develop office systems: A managerial comparison of office system development methodologies, *The Computer Journal, 34*(2), 113-121.

Panko, R. & Sprague, R. (1982). Towards a new framework for office support, in *Proceedings of the ACM SIGOA Conference on Office Information Systems*, pp. 82-92.

Saiedian, H. (1991). An object-based approach to the specification of office entities, in *Computing in the 1990's*, (LNCS #507), New York: Springer-Verlag, pp. 256-263.

Saiedian, H., Farhat, H. & Zand, M. (1992). A specification methodology to support automation of office procedures, *Congressus Numerantium, 85*, 15-32.

Shu, N. (1982). Specification of form processing and business procedures for office automation, *IEEE Transactions on Software Engineering*, SE-8(5), 499-512.

Teger, S. (1983). Factors impacting the evolution of office automation, in *Proceedings of the IEEE*, pp. 503-511.

Tsichritzis, D., ed. (1985). *Office Automation*, New York: Springer-Verlag.

Tsichritzis, D., Rabitti, F., Gibbs, S., Nierstrasz, O. & Hogg, J. (1982). A system for managing structured messages, *IEEE Transactions on Communications*, COM-30(1), 66-73.

Vittal, J. (1980). Active message processing: Messages as messengers, in *Proceedings of the International Symposium on Computer Message Systems*, Vol. IFIP TC-6, Ottawa.

Wong, K. (1990). An Active Message System, unpublished doctoral dissertation, Kansas State University, Manhattan, Kansas.

Woo, C. & Lochovsky, F. (1986). Supporting distributed office problem solving in organizations, *ACM Transactions on Office Information Systems*, *4*(3), 185-204.

Woo, C. & Lochovsky, F. (1987). Viewing communication as a problem solving activity: An enrichment towards supporting cooperative office work, *IEEE-CS Office Knowledge Engineering*, *1*(3), 18-22.