1995

Message Management Systems at Work: Prototypes for Business Communication

Steven. O. Kimbrough
University of Pennsylvania

Scott A. Moore
University of Pennsylvania

Recommended Citation
Kimbrough, S. O., & Moore, S. A. (1995). Message Management Systems at Work: Prototypes for Business Communication. *Journal of Organizational Computing, 5* (2), 83-100. http://dx.doi.org/10.1080/10919399509540244

Message Management Systems at Work: Prototypes for Business Communication

Abstract
In this article, we describe two applications based on a system for office communication that is more flexible and expressive than other systems. This system allows the computerization of tasks that previously required manual intervention because of each task's complexity. The applications, one automating office tasks and the other simulating a bicycle industry, highlight the system's ability to accommodate changes to the communication language. They also highlight the utility of both the formal language used by the system and the inferential model of communications used to interpret the messages.

Keywords
electronic commerce, office automation, speech act theory, communication tools, electronic data interchange, message management

Disciplines
E-Commerce | Entrepreneurial and Small Business Operations | Other Communication | Speech and Rhetorical Studies

Message Management Systems at Work: Prototypes for Business Communication

Scott A. Moore
University of Michigan
samoore@umich.edu

Steven O. Kimbrough
University of Pennsylvania
The Wharton School
sok@grace.wharton.upenn.edu

May 18, 1994

Abstract

We describe two applications based on a system for office communication that is more flexible and expressive than other systems. This system allows the computerization of tasks that previously required manual intervention because of each task's complexity. The applications, one automating office tasks and the other simulating a bicycle industry, highlight the system's ability to accommodate changes to the communication language. They also highlight the utility of both the formal language used by the system and the inferential model of communications used to interpret the messages.

Keywords

electronic commerce, office automation, speech act theory, communication tools, electronic data interchange, message management

Published in WITS special issue of *Journal of Organizational Computing*, 5:2, 83–100. Thanks to Jim Oliver, the referees, and the editors for comments that helped to focus this paper.

1 Introduction

Messaging systems have received much attention in the literature. Many systems have been proposed and their virtues trumpeted. We have done this ourselves [17, 18, 19, 20] with our concepts of *formal languages for business communication* (FLBCs) and *message management systems* (MMSs). The usefulness of each proposed architecture is measured by its effect on the productivity of the people who use applications built upon the system. To begin the investigation of these benefits, we built two prototype applications upon the MMS/FLBC foundation. In this paper we report on the applications and their effect on office work. We find that this architecture provides a more powerful and flexible method for automating a greater range of office tasks than previously was possible. In the next few sections, before describing these two systems, we briefly motivate and describe the messaging system. We conclude with a discussion of related research and future research directions.

2 Background

A messaging system is a tool for conducting business, whether it be office work, transaction processing, inter-firm contracting, or any other commercial activity. It can support varied activities: coordinating the shipment of goods between firms, announcing an upcoming meeting, informing users of a software bug, planning lunch, and so on. Perhaps because of the diversity, messaging systems differ on many attributes. For reasons which shall soon be apparent, this paper focuses on the *formality* of the system's messages. Message formality indicates how much of a message's content is amenable to computer processing. Electronic mail (e-mail) systems pass messages of low formality while electronic data interchange (EDI) systems pass messages of high formality. Each system has its own benefits and drawbacks. E-mail messages are highly expressive but provide little opportunity for computer processing. EDI protocols have limited expressive power but allow the message recipient's computer system to process almost all of its content.

Researchers have merged these two systems so that the benefits of a highly formal system (more computer processing) are available to a highly informal system (that is more expressive). The result of this merger is called a *semi-structured* (by Malone [24]) or *tagged* (by Kimbrough [20]) messaging system. This type of message has a *header* and a *body*. The message's header is computer-processable; the body contains the non-processable mes-

sage content to be read by the recipient. The benefits of using this system have been documented[8, 25, 26, 33]. Limitations are 1) that message processing is usually limited to routing and filtering since the system processes only a portion of the message, and 2) that the language used in the header is *ad hoc*, and 3) the language used in the header is of limited expressiveness (though it does not have to be).

To overcome these limitations we take the opposite tack in merging these two systems. We use a highly expressive, fully computer-processable language that allows the system to intelligently parse, compose, and send messages. In terms of the semi-structured system, the language allows the non-processable body to be put into the processable header. This makes the benefits of a highly *informal* system (more expressive power) available to a highly *formal* system (that allows more computer processing).

3 General Thesis

In this section we shall briefly discuss our reasons for supporting the creation and use of both an MMS and an FLBC. (This argument is more completely laid out in [17].) First, current communication protocols (e.g., EDI) are not expressively powerful or robust under change. As a result most systems (e.g., EDI standards) undergo long development periods to ensure they can express a relatively significant portion of messages companies want to exchange [16]. However, even after this long process, the protocol will need to be revised when additional messages are discovered. This results in numerous software maintenance problems [28]. Additionally, some messages that companies might desire to send would require such drastic changes to the protocol that they will be left out entirely [17, p. 38].

Discourse management is another problem. A message is usually part of a conversation between trading partners. This context can clarify both the message's interpretation and an appropriate response. Although standards for EDI-based negotiations are beginning to be discussed [10], we are a long way from having practical and powerful discourse modeling and management capabilities.

The final problem raised here is that EDI applications tend not to generalize. The trade press is replete with examples of successful applications of EDI to business problems [1, 5, 9, 10, 11, 13]. Implementation of these applications can be difficult. Reports from the field are nearly unanimous in saying that there is a high fixed cost in setting up any innovative EDI application. It is also reported that this investment must largely be repeated,

even for similar applications of EDI [7, 28].

To overcome these difficulties we propose that applications use message management systems (MMSs) that communicate using a formal language for business communication (FLBC). The language can express what the message does, what it is about, and essential elements of its context. The message is not tagged by the formal language description—*the message itself is in the formal language*. A complete discussion of the language is beyond the scope of this paper (we have discussed it in [19] and [20]). Here we simply note the language is recursively defined and fully machine-interpretable, permitting messages of essentially arbitrary complexity and length. Further, the language foundations are in speech act theory (SAT). This theory, though controversial, is widely accepted in linguistics, philosophy of language, and artificial intelligence. Work on SAT began, roughly, with the publication of Austin's *How to Do Things with Words* [3], the text of the lectures he gave for the William James Lectures at Harvard University in 1955. This theory is so named because its adherents assert that to say something is to perform an act. *Act* is being used in a technical and theory-laden, if not altogether clear, way. Briefly, to act is to do something with an intention. Falling down is usually not an action, pulling a lever in a polling booth normally is. The FLBC represents an act a person makes when he communicates.

In contrast with the usual system, an application using the MMS infers its response to a received message. This is in the spirit of an *inferential* theory of communications as opposed to a *decoding* theory.[1] Under a decoding theory, the recipient knows precisely what the speaker intends the recipient to do once the message is decoded. This works well with a limited number of messages but becomes unwieldy with increasing communication complexity. On the other hand, and roughly, under an inferential theory the recipient must first infer what the speaker means and then take the message as a basis of inference for how to act. The recipient might take into account previous messages, who the speaker is, when the message was sent, and what was said (among other things). A message is interpreted, not individually, but in the context of the conversation of which it is a part. Communicating with a formal language under an inferential theory, while endowing a system with more expressive power than otherwise possible, places extraordinary demands on message processing. Using an inferential theory is, in effect, placing a bet that general message handling procedures do exist and can be defined and utilized. Problems with this approach are determining what the message types are and how to define them. SAT answers the first

[1] For a discussion of this, see [30]. A different inferential theory is explained in [4].

and provides hints for the second. Most speech act theories propose a small number of message types and provide general descriptions of how each type differs from the others. We have taken one point of view (closely related to that of Bach & Harnish) and implemented systems that have message handling procedures defined for each message type proposed by the theory. These systems provide one test of the hypothesis that few message types exist, useful distinctions can be made, and procedures can be defined for handling each.

Here is a brief, simple example of the communication process. In the FLBC a simple message consists of a description of the speaker, hearer, illocutionary attitude expressed, a predicate, and the context. It is represented as

msg(Spkr, Hr, IllAtt, Pred, Ctxt, ID)

Illocutionary attitude (`IllAtt`) is a speech act concept but can be simply thought of as what the speaker is doing in sending a message, such as asserting, requesting, or denying something. The predicate (`Pred`) is the content of the message. For example, a person can promise to clean his room or report that his room is clean. These two utterances represent two different illocutionary attitudes (*promising* and *reporting*) in combination with the same predicate (*his room is clean*). The context (`Ctxt`) is other information the speaker thinks might help the hearer understand the speaker's message. The MMS allows responses to messages to vary dependent upon each of these five arguments. For example, suppose that we want to instruct the system that `H` is to do `X` if requested unless the request comes from `mike` in which case `H` will do `not(X)`. The information can be represented in the knowledge base as follows:

process msg(S, H, request, X, Ctxt, ID) if
    S = mike,
    H does not(X).

process msg(S, H, request, X, Ctxt, ID) if
    S ≠ mike,
    H does X.

To see how this might be used, consider the following. If `matt` (or anyone except for `mike`) asks `carrie` to attend a meeting, `carrie` will go to the meeting. If `mike` had asked her to attend, she would not have attended the meeting.

The purpose of this example is not to show two interesting rules (it has probably failed this criterion). It is to demonstrate the following:

> Rules such as these can be very helpful in allowing computers to do some of the work of humans, and
>
> These rules can be expressed easily and naturally.

Universal instantiation (the `X` variable in both procedures and the `S` variable in `(3)`) contributes to the ease with which rules can be composed. The procedure writer does not have to know what is being requested or who (other than `mike`) is making the request—these two rules cover all cases. We recognize that this process of defining appropriate responses will never be completed. The knowledge base will evolve as experience with the system grows, as the vocabulary grows, and as the number of users grows. *A strength of the FLBC is that it encourages and facilitates the composition of these rules and the evolution of the knowledge base.*

The above design decisions address the limitations of current messaging systems (discussed at the beginning of this section). First, changing from the flat structure of a record passing protocol to the recursively defined FLBC increases the expressiveness of the communication language. We submit that this is obvious given that the FLBC is Turing machine equivalent. Second, the FLBC is more robust than current protocols because of both the separation of grammar and vocabulary and the recursive nature of the language. Why is this so? It is certainly possible that a protocol can be appropriately revised for any particular message that needs to be sent; however, what we want is a protocol that can send any message *without* the need for revision. The FLBC provides this capability.

The benefit of the FLBC's expressiveness can possibly be understood more clearly in analogy with movable type.[2] Was it the case that before movable type there were things that could not be printed? No, movable type simply made printing feasible for more books by reducing the printing cost of any particular book. What is the great insight behind movable type? Instead of carving each page separately, printers could take a small number of reusable items (letters) and combine them in many different ways. A benefit of this method is that new words and sentences can be printed using existing pieces instead of requiring each page to be constructed independently of others.

This story is analogous to that of our FLBC (though we are, of course, not comparing the importance of movable type with the importance of

[2]Thanks to Dave Blair for this analogy.

FLBC). A recursively defined language does not allow a machine to represent concepts beyond those that a record passing protocol can represent. It simply makes it easier to represent these concepts by providing a small number of reusable items (in this case, predicates and speech acts) that can be combined in many different ways. If a new concept needs to be expressed, the FLBC allows new predicates to be integrated into existing predicates without disruption.

Third, discourse management is enabled through the definition of a contextual language that provides a framework for more effectively capturing this information. SAT helps structure a conversation because basic discourse structures can be defined based on the illocutionary attitude. For example, a *request* should be followed by an *assertion* or another *request* (see the work in [35]).

Finally, using a universal messaging system and a single grammar increases the generality of messaging applications. The implementation of one system should help considerably with the implementation of the next such system. Procedures for sending messages, receiving messages, and retrieving messages will be the same across applications. Parts of the vocabulary also will be the same across applications. As with any other activity, as programmers gain experience in programming with these tools their proficiency will increase. Conversely, as these tools are improved for one application the benefits will be felt throughout all the applications that use them. Companies will thus have a higher return on improving these tools than on code that is only used by one program.

4 Potential Capabilities

In the previous section we examined the reasons supporting using MMS/FLBC as a basis for a communication system. We now examine how office communications might benefit from this foundation. Two distinct sorts of capabilities are provided by an MMS/FLBC system—one for *saying* and one for *deciding*. The FLBC provides a rich framework in which more and more varied statements can be said formally than otherwise possible while the MMS provides a tool able to make complex decisions about what to do in response to these messages. The following list highlights some capabilities provided by an office communication application based on the MMS/FLBC system.

> *Send unknown number of message types.* It is impossible to foresee before an application is implemented all the message types that might be

needed. This indicates a need for a communication language that can incorporate change easily. The FLBC language is able to express messages of arbitrary complexity, permitting sentence composition with a wide variety of logical operators (well beyond the Boolean operators). Also, the language's strong theoretical basis (in this case, speech act theory), combined with a separation of the grammar and vocabulary, encourages a system that will be robust under change. Thus, the robust and expressive nature of the language contribute to the application's ability to send a large variety of unforeseen messages.

Discuss unknown objects and types of objects. Before an application is implemented, it is impossible to imagine all the subjects of discourse (products, companies, etc.). Thus, the application's vocabulary must be easily expandable. Since a change in the FLBC's vocabulary does not require a change in the grammar, a vocabulary addition is much less difficult than in current systems.

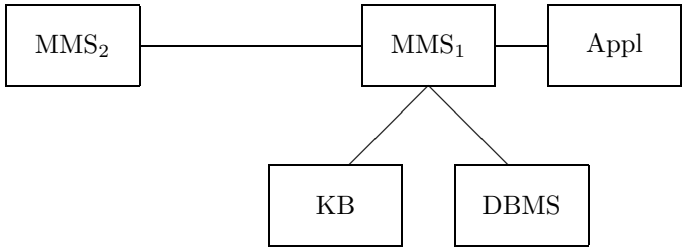
Computerize more office tasks. It is widely recognized that office productivity has not increased at the desired rate over the last several decades. We propose that using the MMS/FLBC combination would make it easier to integrate office operations and computerize tasks that were previously handled manually. Since the message content is machine processable, more information is available to the computer as the basis for inferring an appropriate course of action than with a tagged message system. Further, this is a more complex basis since the language is more expressive than previous systems. This allows a computer to aid in processing and responding to the message.

These capabilities provide a sound basis on which to build an office administration application. Before examining the prototype of such a system we shall discuss the architecture of a system that employs an MMS.

5 Basic Architecture

In this section we discuss the architecture of a system (see Figure 1 on the following page) designed around MMS/FLBC capabilities. Applications (with the aid of its MMS) send and receive messages in an FLBC and intelligently act on the messages using rule-based inference engines (KB). The messages are stored in a data base (DBMS) which is accessible by the MMS. The location of a remote MMS (such as MMS_2 in Figure 1 on the next page)

Figure 1 Basic Architecture



is not constrained. It can be on the same machine or accessible only through a network.

Applications do not communicate directly with one another but use the MMS to handle this process. This helps localize to the MMS the difficulties inherent in sending and receiving messages. One benefit of this protocol is that changes to the communication hardware or software are invisible to any applications that use the MMS.

Because a communicating application must use the facilities provided by the MMS, over time the MMS will accumulate a large data base of messages related to business operations and procedures. To access efficiently any one part of this ever-growing data base of information, the company will work to improve the information retrieval mechanism in the MMS. Since messages are accessible only through the MMS, other applications that use the MMS will also have access to the improved mechanism.

Part of the difficulty of using an FLBC is the flexibility it allows. The cost of this flexibility is increased computational demands. Communication based on decoding and record passing is much less computationally intensive than the model we propose based on inference and a recursively defined language. We propose that the need for flexibility outweighs its cost.

The MMS provides a means for users to send arbitrarily-complex messages; however, this facility is so flexible that it makes it difficult and time-consuming to send simple messages. To manage this flexibility, applications should provide some method of sending oft-repeated messages that hides the FLBC's power and flexibility from the user when not needed.

Because the language is machine processable, the MMS can act on the message given its knowledge about the content of the message, procedures, and other relevant messages. This system design encourages the construction of large, integrative applications that can track complex procedures.

Office administration is such an application. In the next section we examine a prototype.

6 A Prototype: Army Office Communication

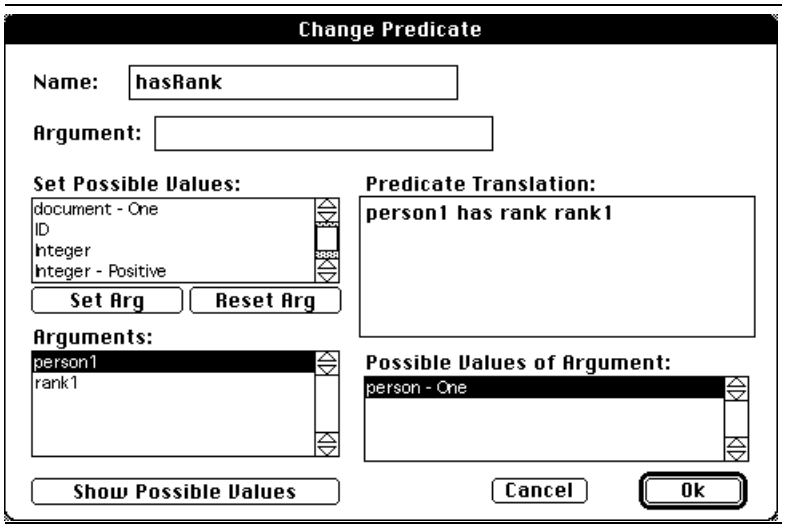
So far, we have stated our hypothesis that many types of electronic communications should be expressed in an FLBC and be managed by an MMS. We then discussed some of the capabilities of a system built on this foundation. We followed this with a discussion of the architecture of an application that uses these facilities. In this section we briefly describe an application we built that is based on the MMS/FLBC foundation.

The daily operations of an army office were observed and several oftensent message types were identified: 1) read/review/comment, 2) requests for an appointment, 3) dissemination of information, 4) staff action, 5) query for information, 6) notice of absence, and 7) statement. (These are referred to as the standard messages.) The army environment provided some advantages that simplified many inferences and made it an attractive application for our first prototype. Clear lines of authority in an army office present opportunities for computerized inferencing on messages. Also, within such an office much information is carried in both the rank and relationship of the individuals involved. These present clear data points on which inferencing can be based.

To this point we have not been as precise as we could have been about the division of responsibilities between the application and the MMS. Before implementing the MMS, we settled on the illocutionary attitudes it could express, the appropriate time representation, the allowable inferences based on the temporal information provided, and the contextual information the MMS could capture. All this information is in the MMS itself. The MMS includes facilities both for creating, sending, receiving, and retrieving all types of messages and for helping the application do the same. Default procedures for handling each illocutionary attitude were defined and added to the MMS knowledge base. The MMS has no user interface—it is invisible to the user.

The army office administration program, on the other hand, is a tool designed to be usable by officers and other workers in an army office. One of the primary purposes of this program is to provide a simple, coherent, and consistent user interface. Standard Macintosh menus and dialog boxes made this step quite easy to achieve. The application itself contains the predicates necessary to create the messages we want to send. The appli-

Figure 2 Defining or Changing a Predicate.



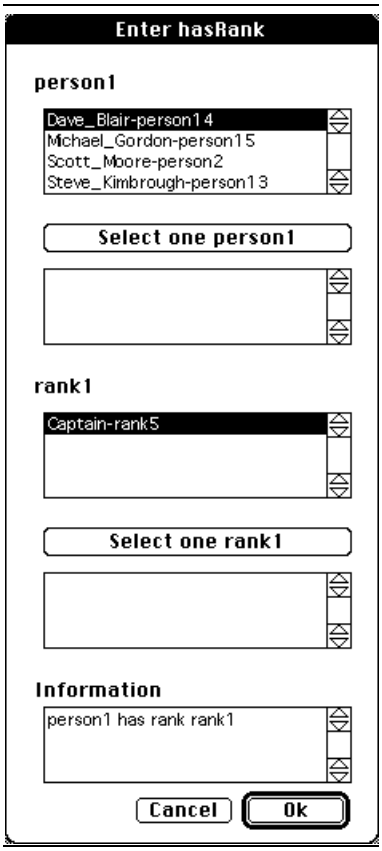
cation knowledge base contains the English translation of each predicate. It also has separate (though integrated) sub-systems for maintaining a person's reminder list, examining and otherwise working with an office schedule, maintaining a list of requests to which other people have not responded, and (of course) sending and receiving messages.

Several features directly relate to capabilities provided by the MMS/FLBC foundation. The program facilitates the process of adding to the FLBC vocabulary. For example, menu choices are provided for defining predicates. (Predicates are the carriers of the *content* of a message; i.e., what is asserted, requested, etc.) Defining a predicate involves giving it a name, giving each of its arguments a name, defining the allowable type(s) for each argument, and the English translation of the predicate. For example, consider `hasRank(x,y)`. As defined in Figure 2 on this page, this predicate is named `hasRank` and can be translated *x has rank y*. Further, argument #1 is called `x`, argument #2 is called `y`, `x` can be of type *person*, and `y` can be of type *rank*,

A menu choice is also provided for adding objects to the vocabulary. *Objects* are the entities to which predicates can refer. For example, `hasRank(p(12),r(6))` refers to two objects: `p(12)` and `r(6)`. We know from the definition that `p(12)` is a *person* object and `r(6)` is a *rank* object.

This application not only allows the user to enter additional objects as needed but also provides a dialog box for defining new object types.

Figure 3 A System-Generated Dialog.



The creation of new objects is a routine capability provided even by simple database programs. For example, when a person is hired a person object is created using information gathered in a dialog box. This application goes beyond this by providing a tool for creating new object types. The user can define what types of objects are possible message topics. The application uses this information to generate (at run time) the dialog box for entering objects of that type (see Figure 3 on the current page).

As mentioned in §5 part of the difficulty in using this type of communication system is the flexibility it allows. To combat the complexity dialogs are defined for each standard message listed at the beginning of this section. These dialog boxes employ popup menus, scrolling lists, and text fields to elicit information from the user. When the user is done specifying information and clicks on the *Send* button, the application composes

the appropriate FLBC message and instructs the MMS to send the message to the addressees. This dialog box is also generated automatically from a description of the standard message.

Although at times the user must be shielded from the system's flexibility, he can access it when needed. A dialog box for generating an arbitrary message is callable from a separate menu item. If none of the standard messages are appropriate, the user can employ this general message dialog box to specify the message. This dialog box allows the iteration of illocutionary attitudes: a speaker can *request* that someone *request* that someone else *inform* that *Z*. Iteration of messages is difficult for a record-passing protocol (e.g., EDI) to express.

Finally, since the MMS/FLBC system is able to use all types of information as a basis for inference, the application collects much information from the user as she performs her job. The system knows about each person's previously-sent messages, reminder list, list of unfulfilled requests, and daily schedule. All of this knowledge is available when the MMS logs a message for a user. If it is an appointment request, the application checks the schedule and notifies the sender of the message if the person will be absent on that day. If it is a query for information, the system answers it if it can—if it cannot, it presents the question to the message recipient. If it is a statement in response to a question, the system will delete the question from the list of unfulfilled requests. In short, the system is able to make and exploit a rich and subtle set of inferences.

In this section we described an application's ability to send arbitrary and standard messages about user-defined predicates and objects. These messages are processed by the application or MMS if possible and forwarded to a person for assistance only when necessary. We next look at another application of an MMS.

7 A Demonstration: A Bicycle Shop

We have modeled a simplified bicycle industry that consists of retail customers, retail shops, bicycle manufacturers, and parts suppliers (see Figure 4 on the following page). The retail customers can assemble a bicycle with the assistance of a graphical user interface (Figure 5 on the next page).[3] The terminal that runs this program can either be in the retail shop or be remotely connected to the shop. The retail shop is electronically connected to the manufacturers who are electronically connected to the parts suppliers.

[3]Thanks to Steve Kirk for writing the code for the bicycle assembly routine.

Figure 4 Sending a Message.

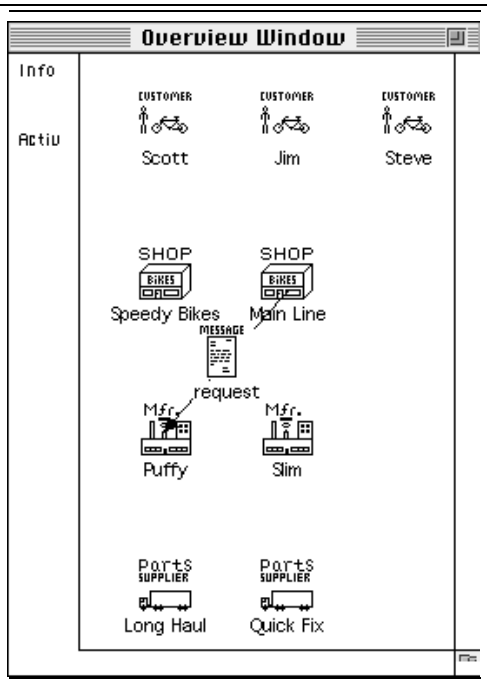


Figure 5 Assembling a Bicycle.

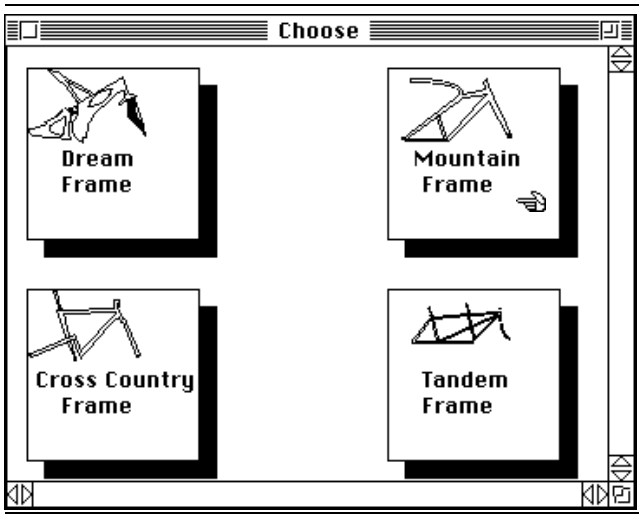
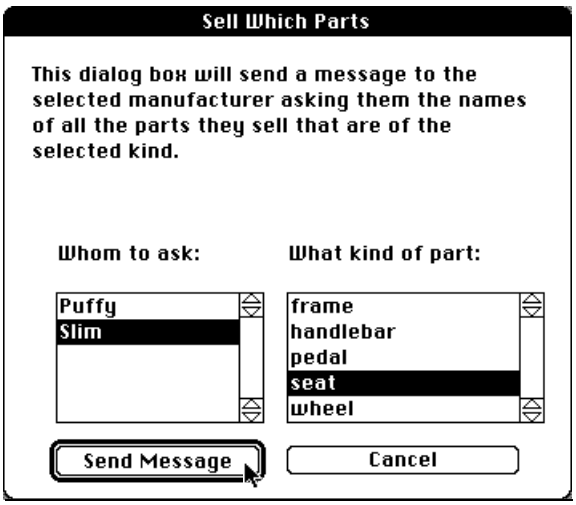


Figure 6 Entering information.



Each of the industry members fulfills its expected role in getting the bicycle to the customer after the order is placed; however, the focus of the demonstration is the messages each is able to send. Each is able to send and process orders and inquiries about inventory. Additionally, the customer is able to check on availability of the specified parts and check the store's preferences about part compatibility. Also, the shop can ask manufacturers about the kinds of parts they sell. Each industry member is able to automatically process the messages it receives.

Many other kinds of messages are sent within this system. For example, in Figure 4 on the preceding page the customer *Scott* sent a request for part availability to the shop *Main Line*. The shop did not have any parts available so it sent a request for information to the manufacturer *Puffy* (the message shown in the figure). The manufacturer checks its stock and sends a message back to the shop. The shop consolidates the answers it receives from the various manufacturers, and then forwards its own answer on to *Scott*. Note that no manual intervention on the part of the shop or manufacturer was required because the MMSs of both the shop and the manufacturer knew how to handle requests for availability. This is a relatively common message type. Both companies realized the benefits from automating the process of handling it.

The shops are also able to send messages. In Figure 6 on the current page the shop selects items from two scrolling lists and clicks the *Send Message* button. The application formats an FLBC message, forwards it to the MMS,

and the MMS sends the question to the proper addressee. When a response is received by the MMS, the message is forwarded to the correct application whereupon the application determines how to best handle the response. In this case an English message is displayed.

Note that many of these message types are similar to those sent within an EDI system. One of the purposes of this demonstration is to show how EDI-like messages can be handled by an MMS/ FLBC system; however, it is also meant to show how messages beyond the scope of EDI messages can (and should) be integrated into such a system.

We built this system in Prolog on the Macintosh.[4] The data needed by both the MMSs and the applications are stored in an SQL database. All the programs are able to access the data with SQL commands embedded in the Prolog code. *The MMS used in this demonstration is virtually identical to the one used for the office administration prototype.* The only differences are the usual code changes made as time passes and changing the low-level calls so that they called an SQL database instead of a Prolog database.

8 Related Research

Using electronic messages to manage office activities is not a new idea; however, significant differences do exist between the system described in this thesis and those described by other researchers. These differences center around the expressiveness of the FLBC and the inferential nature of the communication process. In this section I highlight some of these systems and differences.

Comparing the FLBC with Winograd and Flores's *Coordinator* system [12, 33], several differences become apparent though some similarities exist. The *Coordinator* explicitly recognizes that a message is a speech act and is best understood as part of a larger conversation. The designers also proposed that messages are typically not sent merely to convey information but to do something—to request, to offer, to promise, to decline, etc. Messages are classified according to the action they are meant to perform, similar to the categorization of a statement according to the author's attitude. Similarly, the context of a message in the *Coordinator* is the conversation of which it is a part. However, their system is more of a tagged-message system. The message is classified according to their scheme but the content is expressed separately from this classification.

[4] All of the systems discussed in this paper are written in Quintus/LPA MacProlog.

Comparing the FLBC with Malone et. al's *Information Lens* system [25], the classification scheme used is similar to that employed in the *Coordinator*. In this case the scheme is more pragmatic than semantic, based on some sociological studies they did on information sharing in organizations. Each message is classified in two separate ways: 1) the action the message represents (e.g., action request, notification, or commitment), and 2) the purpose of the message (e.g., a notification can be a software release, a publication announcement, a network discussion item, etc.). This system is also a tagged-message system in which the classification is completely *ad hoc*. In fact, each organization is encouraged to design its own classification scheme.

Pollock's *ISCREEN* system [26] is a message *filtering* system that can, under the control of rules defined by the user, direct the system as to how it should respond to incoming messages. The actions the system can take are basically limited to forwarding, filing, and deleting messages. Also, these rules can only interpret a simple description of a message's content and not the content of the message itself. There is no fundamental recourse to speech act theory.

Tsichritzis's system [31] for managing structured messages has goals similar to the proposed system. Users can "1) file and retrieve messages. . . , 2) locate messages. . . , and 3) query and obtain data present in messages. . . [U]sers can specify procedures which: 1) coordinate messages, i.e., act only when a related set of messages has been assembled; 2) modify and create messages, 3) file messages in dossiers; and 4) automatically forward received messages to other stations according to their contents." [31, p. 66] This system sends and receives fully-formal messages. This system seems to be a direct response to what Tsichritzis sees as the faulty practice of using a tagged-message system. The language he uses is of limited expressiveness and has more in common with EDI than the FLBC.

Chang and Leung's *KMMS* [8] focuses on the problem of junk mail. It attempts to reduce the amount of electronic mail a person must read by providing a linguistic message filter and user-defined alerter rules. For current purposes, this system has several limitations. The structured language used is *ad hoc* and is of limited expressiveness (i.e., it is non-recursive). The rules used by the system are classic first order logic, so all the usual problems arise in the rule base (presence of useless rules, conflicting rules, etc.)

Woo and Chang's implementation [35] is a set of communication tools for helping people during negotiations. Their work is based on Ballmer and Brennenstuhl's extensive speech act classification [6]. Their theory of communication precisely lays out the allowable speech acts that can be made at each step in a conversation. This conversation structure is a type of

contextual knowledge that has not yet been integrated into the FLBC. The messages are not fully formal but are more structured than the messages used by *Information Lens*. This allows some automatic processing of messages. Though Woo and Chang have a theoretical basis for their work, and though their system is useful, many opportunities exist for exploiting an inference based communication system which is not done in their system.

Turner and Cullingford's *JUDIS* system [32] is based on *MOPs* (memory organization packets) [29] and, more specifically, *conversation MOPs* [15]. Schank defines a MOP as "[i]nformation about how memory structures are ordinarily linked in frequently occurring combinations" [29, p. 83]. In JUDIS these are used to structure the flow of conversation by taking account of both the intention of the speaker and the conventional rules of conversation. Their model of communication is quite complex and based on cognitively plausible principles. The goals of this system are quite similar to the proposed system though their methodology is quite different.

Woo proposes SACT, a speech act theory based communication system for automation some communication tasks [34]. This system uses a language less expressive than the FLBC though it does use a speech act based framework. Also, he does not allow the meaning of an utterance to differ from its surface representation: "there is no ambiguity in the meaning of the sentence when the category *[illocutionary attitude]* is known." [34, p. 91] This limits the flexibility of the communication system but also provides a less ambiguous basis for interpretation for the communication system. Only experience will tell whether or not the loss of expressiveness outweighs the loss in ambiguity.

The *SAMPO* methodology [2, 23] is used to model office communication and the effects of this communication on commitments. This methodology is based on speech act theory and has much in common with the work presented here. However, the focus of this work seems to be on *modeling* office work and not *supporting* office work. The capabilities of their system can act as a measuring stick for our system—if their system can model it, then our system should be able to support it. Another difference between the two systems is that their communication model seems to be more of a decoding-based model that limits the possible interpretations of a message by a recipient. As stated previously, we prefer to use an inference-based communication model.

The *diplans* language [14] is a formal graphical representation of a plan, involving both human and computer participants. The graph represents those activities that must be completed by people and computers throughout an organization in the course of the completion of some task. It is used

to manage the coordination needed to efficiently and effectively complete the described task. The *diplans* language is a relatively complex graphical language but, at the same time, a simple language of relatively limited expressiveness. The communication language, though not fully described, also seems to be relatively simple, as it only needs to indicate the status of a stage in a plan.

Lee's work [21, 22, 27] is about bureaucracies as systems for monitoring and controlling obligations and permissions. His system could also be used as a test of our system. If it models a bureaucracy that acts in a certain way, then the proposed system should be able to provide some level of computerized support of the people in the bureaucracy. His work supports the contention we made [18] that deontic reasoning is not a tangential capability of organizations but is primary.

9 Discussion and Further Research Opportunities

This paper briefly describes two systems whose foundations are the MMS/FLBC concepts. We need to emphasize that the calls to the MMS in each system are identical—the data for each program could be stored in the same database if so desired. The routines used to retrieve and search for data could be—and should be—the same in both systems. The purpose of showing two systems is to demonstrate that the FLBC/MMS concepts are flexible enough to handle different application requirements while maintaining the same functionality and programming interface.

Much interesting research and field testing remains. We must create a method for the end user so she is easily able to define procedures for responding to messages. We are attempting to integrate defeasible reasoning about temporal and deontic information into the MMS and applications [18]. Finally, we are creating a method of describing procedures so that the description provides the basis for tracking business processes.

We realize much work needs to be done before the MMS/FLBC concepts are commercially viable. However, we think that the two systems discussed in this paper are encouraging signs that these concepts have potential for widespread usage in the coming era of "anything, anytime, anywhere" communication.

References

[1] Linda G. Allen. Small-town company talks big savings, better service with EDI. *Automation*, 38(7):56, July 1991.

[2] Esa Aurämaki, Erkki Lehtinen, and Kalle Lyytinen. A speech-act-based office modeling approach. *ACM Transactions on Office Information Systems*, 6(2):126–152, April 1988.

[3] John L. Austin. *How To Do Things With Words*. Harvard University Press, 2nd edition, 1975.

[4] Kent Bach and Robert M. Harnish. *Linguistic Communication and Speech Acts*. MIT Press, 1979.

[5] Carol Baker. EDI in business. *Accountancy*, 107(1172):121–124, April 1991.

[6] T. Ballmer and W. Brennenstuhl. *Speech act classification*. Springer-Verlag, 1981.

[7] Collin Canright. Customization comes to EDI. *Bank Management*, 67(10):59–62, October 1991.

[8] Shi-Kuo Chang and L. Leung. A knowledge-based message management system. *ACM Transactions on Office Information Systems*, 5(3):213–236, July 1987.

[9] Jeanette Clickunbroomer. As essential as a telephone. *Global Trade*, 111(9):41, 50, September 1991.

[10] Wayne Eckerson. Pioneering users moving to faster methods of EDI. *Network World*, 8(18):1, 1991.

[11] Barnaby J. Feder. Moving the Pampers faster cuts everyone's costs. *New York Times, business section*, page 5, July 14, 1991.

[12] Fernando Flores, Michael Graves, Brad Hartfield, and Terry Winograd. Computer systems and the design of organizational interaction. *ACM Transactions on Office Information Systems*, 6(2):153–172, April 1988.

[13] Derek Harris. Fast orders help traders net profits. *Accountancy*, pages 104–105, November 1991.

[14] Anatol W. Holt. Diplans: A new language for the study and implementation of coordination. *ACM Transactions on Office Information Systems*, 6(2):109–125, April 1988.

[15] Kathy Kellermann, Scott Broetzmann, Tae-Seop Lim, and Kenji Kitao. The conversation MOP: Scenes in the stream of discourse. *Discourse Processes*, 12:27–61, 1989.

[16] Paul Kimberley. *Electronic Data Interchange*. McGraw-Hill Inc., New York City, 1991.

[17] Steven O. Kimbrough and Scott A. Moore. Message management systems: Concepts, motivations, and strategic effects. *Journal of Management Information Systems*, 9(2):29–52, Fall 1992.

[18] Steven O. Kimbrough and Scott A. Moore. On obligation, time, and defeasibility in systems for electronic commerce. In J.F. Nunamaker, Jr., editor, *Proceedings of the Hawaii International Conference on System Sciences*, volume III, pages 493–502. University of Hawaii, IEEE Computer Society Press, 1993.

[19] Steven O. Kimbrough, Sashidhar Reddi, and Michael Thornburg. On messaging with semantic access in an office environment. Working papers series, University of Pennsylvania, Wharton School, Decision Sciences Department, December 1989.

[20] Steven O. Kimbrough and Michael Thornburg. On semantically-accessible messaging in an office environment. In *Proceedings of the Twenty-Second Hawaii International Conference on System Sciences*. University of Hawaii, IEEE Computer Press, 1989.

[21] Ronald M. Lee. Bureaucracies as deontic systems. *ACM Transactions on Office Information Systems*, 6(2):87–108, April 1988.

[22] Ronald M. Lee, S.D. Dewitz, and K.T. Chen. AI and global EDI. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, volume IV, pages 182–191. University of Hawaii, IEEE Computer Society Press, 1991.

[23] Erkki Lehtinen and Kalle Lyytinen. Action based model of information system. *Information Systems*, 11(4):299–317, 1986.

[24] Thomas W. Malone, Kenneth R. Grant, Kum-Yew Lai, Ramana Rao, and David Rosenblitt. Semistructured messages are surprisingly useful for computer-supported coordination. *ACM Transactions on Office Information Systems*, 5(2):115–131, April 1987.

[25] Thomas W. Malone, Kenneth R. Grant, Franklyn A. Turbak, Stephen A. Brobst, and Michael D. Cohen. Intelligent information-sharing systems. *Communications of the ACM*, 30(5):390–402, May 1987.

[26] Stephen Pollock. A rule-based message filtering system. *ACM Transactions on Office Information Systems*, 6(3):232–254, July 1988.

[27] Young U. Ryu and Ronald M. Lee. Defeasible deontic reasoning: A logic programming model. In J.J. Ch. Meyer and R.J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*, chapter 9, pages 1–17. John Wiley & Sons Ltd., 1993.

[28] Salvatore Salamone. EDI: Bottom-line booster or budget-breaker? *Network World*, 8(13):1ff, April 1, 1991.

[29] Roger C. Schank. *Dynamic Memory*. Cambridge University Press, 1982.

[30] Dan Sperber and Deirdre Wilson. *Relevance: Communication and Cognition*. Harvard University Press, 1988.

[31] Dennis Tsichritzis, Fausto A. Rabitti, Simon Gibbs, Oscar Nierstrasz, and John Hogg. A system for managing structured messages. *IEEE Transactions on Communications*, 30(1):66–73, January 1982.

[32] Elise H. Turner and Richard E. Cullingford. Using conversation MOPs in natural language interfaces. *Discourse Processes*, 12:63–90, 1989.

[33] Terry Winograd and Carlos F. Flores. *Understanding Computers and Cognition*. Ablex Publishing Corp., Norwood, NJ, 1986.

[34] Carson C. Woo. SACT: A tool for automating semi-structured organizational communication. In Frederick Lochovsky and Robert B. Allen, editors, *Proceedings of Conference on Office Information Systems*, pages 89–98. ACM SIGOIS & IEEECS TC-OA, ACM Press, April 1990.

[35] Carson C. Woo and Man Kit Chang. An approach to facilitate the automation of semistructured and recurring negotiations in organizations. *Journal of Organizational Computing*, 2(1):47–76, 1992.