Semi-Structured Messages are Surprisingly Useful for Computer-Supported Coordination

Thomas W. Malone
Kenneth Grant
Kum-Yew Lai
Ramana Rao
David Rosenblitt

90s: 86-026

December, 1986

CISR WP# 148
Sloan WP# 1851-86

Published in the Proceedings of the Conference on Computer-Supported Cooperative Work, Austin, Texas, 3-5 December 1986

Management in the 1990s
Sloan School of Management
Massachusetts Institute of Technology

Management in the 1990s is an industry and governmental agency supported research program. Its aim is to develop a better understanding of the managerial issues of the 1990s and how to deal most effectively with them, particularly as these issues revolve around anticipated advances in Information Technology.

Assisting the work of the Sloan School scholars with financial support and as working partners in research are:

American Express Travel Related Services Company
Arthur Young and Company
British Petroleum Company, p.l.c.
BellSouth Corporation
Digital Equipment Corporation
Eastman Kodak Company
General Motors Corporation
International Computers, Ltd.
MCI Communications Corporation
United States Internal Revenue Service

Semi-Structured Messages are Surprisingly Useful for Computer-Supported Coordination

Thomas W. Malone, Kenneth R. Grant, Kum-Yew Lai, Ramana Rao, and David Rosenblitt

Massachusetts Institute of Technology

Abstract

This paper argues that using a set of semi-structured message templates is surprisingly helpful in designing a variety of computer-based communication and coordination systems. Semi-structured messages can help provide automatic aids for: (1) composing messages to be sent, (2) selecting, sorting, and prioritizing messages that are received, (3) responding automatically to some messages, and (4) suggesting likely responses to other messages. The use of these capabilities is illustrated in a range of applications including electronic mail, computer conferencing, calendar management, and task tracking. The applications show how ideas from artificial intelligence (such as inheritance and production rules) and ideas from user interface design (such as interactive graphical editors) can be combined in novel ways for dealing with semi-structured messages. The final part of the paper discusses how communities can evolve a useful set of message type definitions.

Semi-Structured Messages are Surprisingly Useful for Computer-Supported Coordination

The goal of this paper is to articulate one of the major lessons learned so far from our design and use of the Information Lens, an intelligent system for information sharing in organizations (Malone, Grant, Turbak, Brobst, & Cohen, in press). The simple idea of using semi-structured message templates turns out to be suprisingly powerful, with implications for designing a variety of group communication and coordination systems. We first describe how this idea simplifies the design of general capabilities for composing and processing messages. Then we show how these general capabilities have helped us develop not only our original information sharing application, but also simple applications for a variety of other purposes such as task tracking, calendar management, and computer conferencing. Finally, we discuss several general issues about how a community defines its message types.

Most previous systems for supporting group work can be thought of as belonging to one of two categories. In one category, are forms processing, calendar management, and other systems that contain a great deal of formalized knowledge about the application domains they are intended to support (e.g., Greif, 1982; Ellis & Bernal, 1982; Fox et al, 1983; Sluizer & Cashman, 1985). In the other category are electronic mail, computer conferencing, and hypertext systems that provide some very general representational tools (such as topic structures in computer conferences or embedded links between documents) and very little other formal knowledge about their domains (e.g., Engelbart & English, 1968; Hiltz & Turoff, 1978; Trigg, Suchman, & Halasz, 1986). By using semi-structured message templates, we are able to achieve an intermediate--but variable--level of formalization that can flexibly accommodate as much or as little structure as is useful for a particular kind of communication.

Semi-structured messages

We define "semi-structured messages" as messages of identifiable types with each type containing a known set of fields, but with some of the fields containing unstructructured text or other information. For example, seminar announcements can be structured as templates that include fields for "time", "place", "speaker", and "topic", along with additional text describing the content of the talk (e.g., Goldstein, 1980). There are several reasons why this idea is important:

(a) *Semi-structured messages enable computers to automatically process a much wider range of information than would otherwise be possible.* By letting people compose messages that already have much of their essential information structured in fields, we eliminate the need for any kind of automatic parsing or understanding of free text messages while still representing enough information to allow quite sophisticated processing of the messages.

(b) *Semi-structured messages allow people to communicate non-routine information without the constraints of a rigid structure.* By letting people include unstructured text in some fields and use other fields in non-standard ways, we greatly increase the flexibility and usefulness of our systems. Unlike in rigid forms-based systems, unusual situations do not bring the system to halt. Instead, such situations merely reduce the usefulness of the automated support and depend, for their resolution, on the attention of human users.

(c) *Much of the processing people already do reflects a set of semi-structured message types.* It is a common observation that routine information processing in organizations is often based on structured forms. Even when structured forms are not used, we found in our informal studies (Malone et al, in press) that people often describe their processes for filtering information according to categories of documents being filtered (e.g., *This is a brochure advertising a seminar. I usually throw these away unless the title intrigues me or unless it looks like a brochure I could use as a model for the ones I write.* -- paraphrased comments of a research center administrator in our informal study).

(d) *Even if no automatic processing of messages were involved, providing a set of semi-structured message templates to the authors of messages would often be helpful.* Two of the people in our informal interviews mentioned simple examples of this phenomenon: one remarked about how helpful it would be if any memo requesting some kind of action included, in a prominent place, the deadline by which the action needed to be taken; a second commented about how wonderful it would be if all the meeting invitations he received included a field about why he was supposed to be there. We will see below how message templates can be provided in a flexible way that encourages, but does not require, their use.

(e) *Semi-structured messages simplify the design of systems that can be incrementally enhanced and adopted.* The initial introduction and later evolution of a group communication system can be much easier if the process can occur as a series of small changes, each of which has the following properties: (a) individual users can continue to use their existing system with no change if they so desire, (b) individual users who make small changes receive some immediate benefit, and (c) groups of users who adopt the changes receive additional benefits beyond the individual benefits. While semi-structured messages are not necessary for building such incrementally expandable systems, we will see below how they facilitate this process.

In addition to these general reasons why semi-structured messages are desirable, we will also see below how their use can be simplified by: (a) *arranging the message types in a frame inheritance lattice* so that specific message types can "inherit" properties from more general types they resemble, and *(b) using a consistent set of display-oriented editors* for composing messages, constructing message processing rules, and defining new message templates.

Example: The Information Lens system

In order to illustrate our discussion of features made possible by semi-structured messages we will first briefly describe our implementation of the Information Lens, an intelligent system for information sharing in organizations (see Malone et al, 1986). We define the information sharing problem as one of disseminating information to the people who will find it useful without distracting others who will find no value in its contents. This problem is likely to become increasingly important as it becomes both technically and economically feasible to send electronic messages and other documents to large numbers of possible recipients. First, it is already a common experience in mature computer-based messaging communities for people to feel flooded with large quantities of electronic "junk mail" (e.g., Denning, 1982; Hiltz & Turoff, 1985). At the same time, it is also a common experience for people to be ignorant of facts that would facilitate their work and that are known elsewhere in their organization. The Information Lens helps people solve both these problems: it helps people filter, sort, and prioritize messages that are already addressed to them, and it also helps them find useful messages they would not otherwise have received.

More specifically, the Lens system provides four important optional capabilities, in addition to the usual capabilities of an electronic mail system: (1) Senders can compose their messages using

structured templates that suggest the kinds of information to be included in the message and likely alternatives for each kind of information; (2) Receivers can specify rules to automatically filter and classify their incoming messages into folders based on the same dimensions used by senders in constructing messages; (3) Senders can include as an addressee of a message, in addition to specific individuals or distribution lists, a special mailbox (currently named "Anyone") to indicate that the sender is willing to have this message automatically redistributed to anyone else who might be interested; and (4) Receivers can specify rules that find and show messages addressed to "Anyone" that the receiver would not otherwise have seen.

Messages that include "Anyone" as an addressee are delivered by the existing mail server directly to the explicit addressees as well as to an automatic mail sorter that runs on a workstation and periodically retrieves messages from the special mailbox. This automatic mail sorter then, in turn, sends the messages to any additional recipients whose rules select them. Messages can thus be selectively disseminated only to people who are likely to be interested in them. When, for reasons of confidentiality, it is necessary to restrict the potential recipients, messages can be addressed to "Anyone -in- <distribution list name>" and the messages will then be distributed only to members of the distribution list whose rules select them.

This framework supports many kinds of information sharing in addition to straightforward electronic mail. For example, we have recently begun receiving a daily on-line feed of *New York Times* articles (via the system described by Gifford, Baldwin, Berlin, & Lucassen, 1985). These articles already contain fields such as subject, priority, and author, and the users of our system can specify rules based on these fields for selecting and sorting news articles just like the rules for sorting any other kind of message.

Features made possible by semi-structured messages

As noted above, the use of a set of semi-structured message templates simplifies the original composition and subsequent processing of messages. In this section, we list several general features of this type and illustrate them with examples from our original implementation of the Information Lens system (Malone et al, 1986) and our subsequent enhancements to it.

Some of the features we will describe (such as automatic processing of incoming messages) might, in principle, be provided without using structured messages, if robust and general natural language processing capabilities were available. The required capabilities, however, still appear to be

significantly beyond the current state of the art. As natural language parsers improve, they can be used to parse more and more kinds of unstructured documents into templates like those we describe. Then the kinds of knowledge represented in our system will still be necessary, and it will become useful for a much wider range of documents. In a few cases, such as the automatic aids for constructing messages, the simple user interfaces that are possible with structured messages appear to be "cleaner" and probably easier and more efficient to use than what could be done with natural language interfaces to unstructured text.

Automatic aids for constructing messages

The presence of a set of structured message types makes possible a variety of automatic aids for constructing messages. For example, the templates provided by the Information Lens system for each message type include specialized information about each field. Figure 1 illustrates the highly graphical interaction through which users can construct messages using these templates. After selecting a field of a message by pointing with a mouse, the user can point with the mouse again to see the field's default value, an explanation of the field's purpose, or a list of likely alternatives for filling in the field. If the user selects one of these alternatives, that value is automatically inserted in the message text.

By providing a wealth of domain-specific knowledge about the default and alternative values for particular types of messages, the system can make the construction of some messages much easier. For example, Figure 2 shows a regular weekly meeting announcement with default values already filled in for most of their fields. These values can be computed based on other information in the system (e.g., the alternatives for the "To" field might be a list of people to whom other recent messages have been addressed).

Users who do not want to take advantage of these message construction aids can always select the most general message type (*message*) and they can always edit any fields directly at any time using the built-in, display-oriented text editor. For example, the user can add as much free text as desired in the text field of the message. We expect, however, that the added convenience provided to the senders by semi-structured templates will be a significant incentive for senders to use templates in constructing their messages.

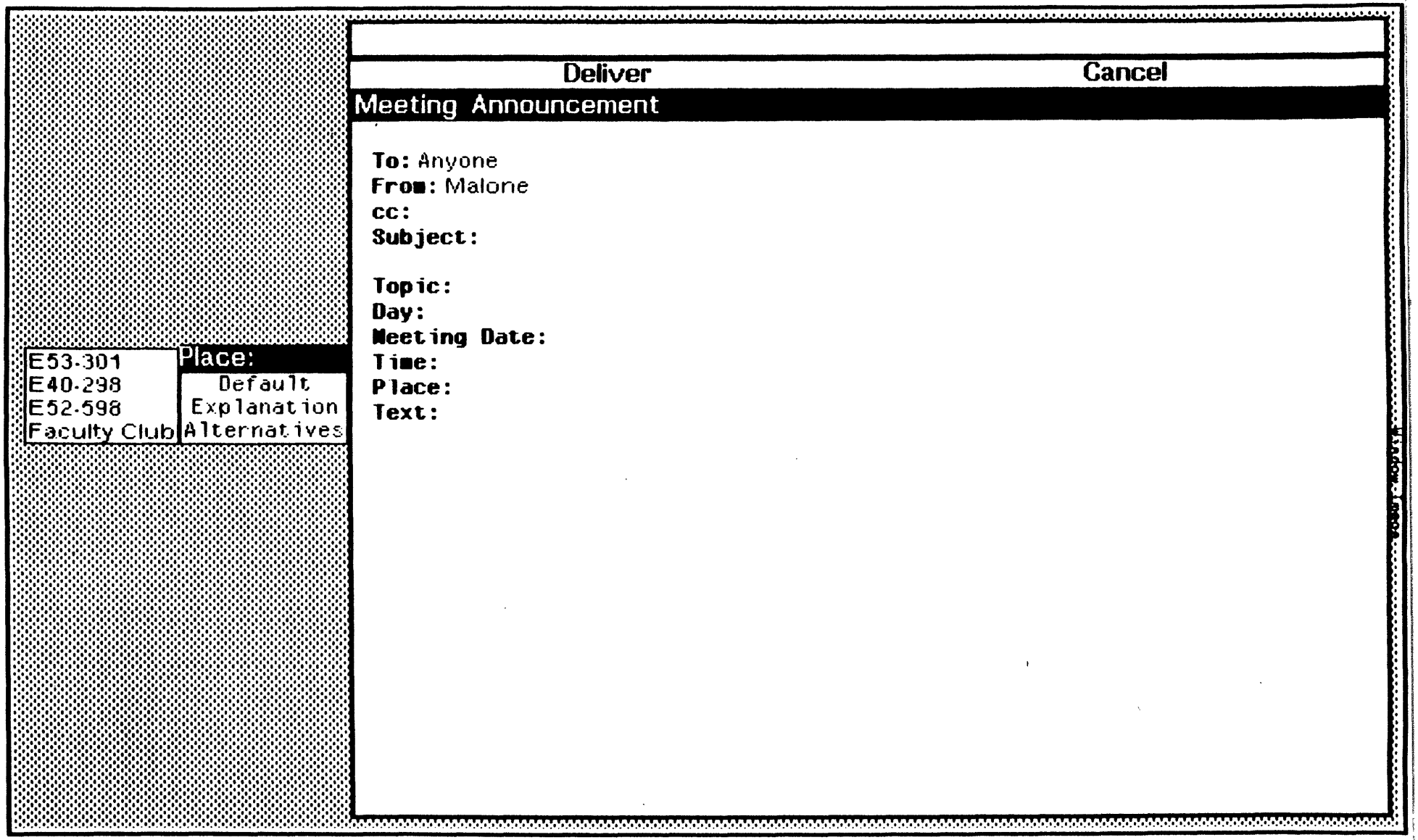

Figure 1

Messages are composed with a display-oriented editor and templates that have pop-up menus associated with the template fields.

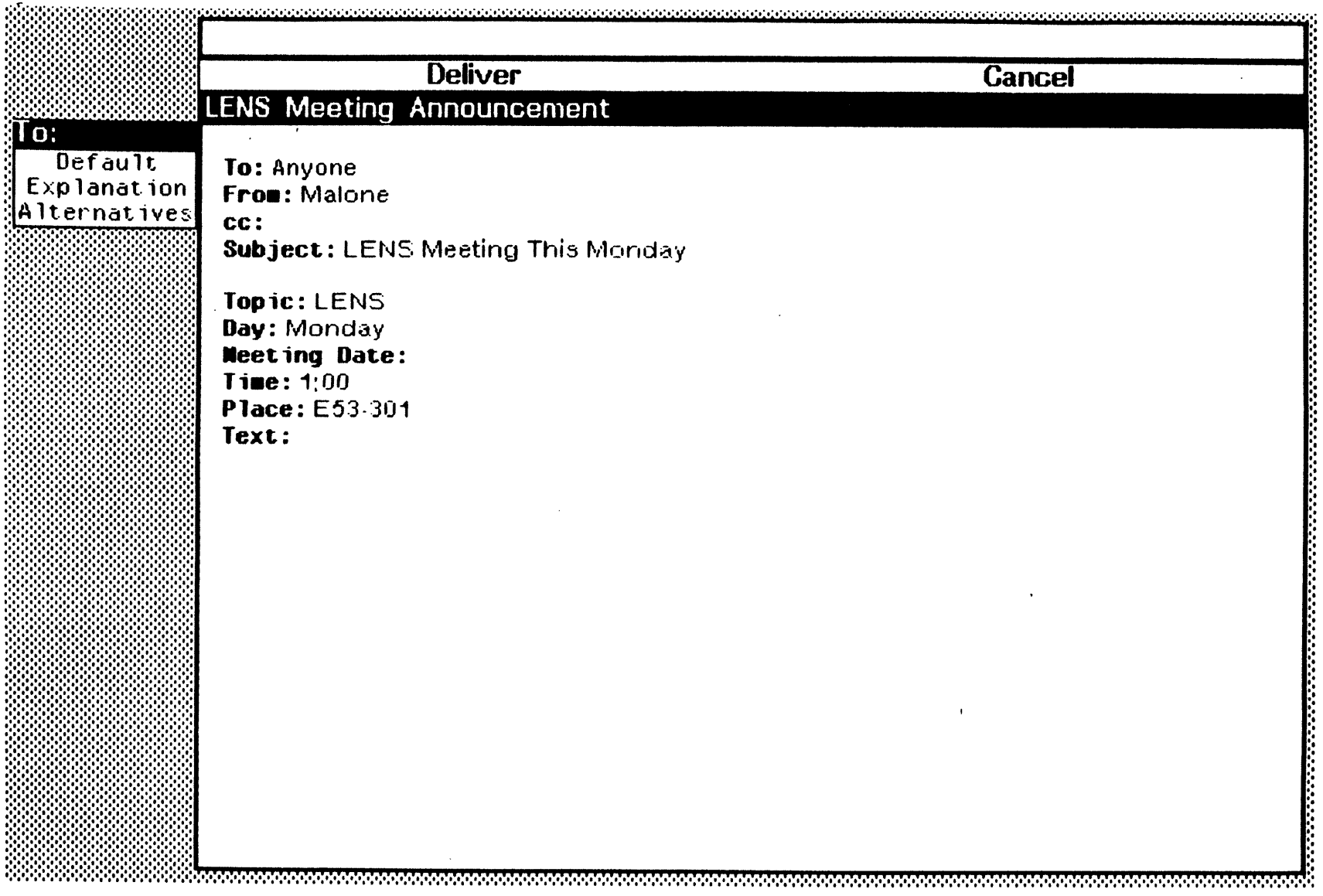

Figure 2

Some templates already have a number of default values filled in for different fields.

Rules for automatically processing messages

Just as the structure of messages simplifies the process of composing messages, it also simplifies the process of constructing rules for processing messages. For instance, Figure 3 shows an example of the display-oriented editor used to construct rules in the Information Lens system. This editor uses rule templates that are based on the same message types as those used for message construction, and it uses a similar interaction style with menus available for defaults, alternatives, and explanations. We expect that this template-based graphical rule construction will be much easier for inexperienced computer users than more conventional rule or query languages. For example, the users of most typical database retrieval systems must already know the structure of the database fields and their plausible values in order to construct queries. Users of our system have all this information immediately available and integrated into the rule-construction tools (Tou, Williams, Fikes, Henderson, & Malone, 1982).

Typical rule actions move messages to specific folders (Figure 4a), delete messages (Figure 4b), or select messages addressed to "Anyone" that the user would not otherwise have seen (Figure 4c). By combining conditions within and between fields, users can construct arbitrary Boolean queries. More interestingly, users can also construct elaborate multi-step reasoning chains by having some rules set *characteristics* of messages and then having other rules test these characteristics (Figures 4d & 4e; see Malone et al, in press, for details).

Our early experience with the system suggested the importance of being able to give users some control over the interactions between rules (e.g., having certain rules fire only if no other rules have fired on a message) and of being able to explain to users why certain messages were processed as they were. Accordingly, we have recently added several simple features to the rule system. First, rules are applied in order within each rule set and the rules pertaining to a specific message type are always applied before the rules inherited from more general message types (see below). Also, rules that take actions (such as moving or deleting a message), always set a characteristic of the message (such as "MOVED" or "DELETED"). Thus subsequent rules can include conditions such as "the message has not yet been moved or deleted" (see Figure 4e). Finally, in order to help users understand and modify their rules, a simple explanation capability allows users to see a history of the rules that fired on a given message.

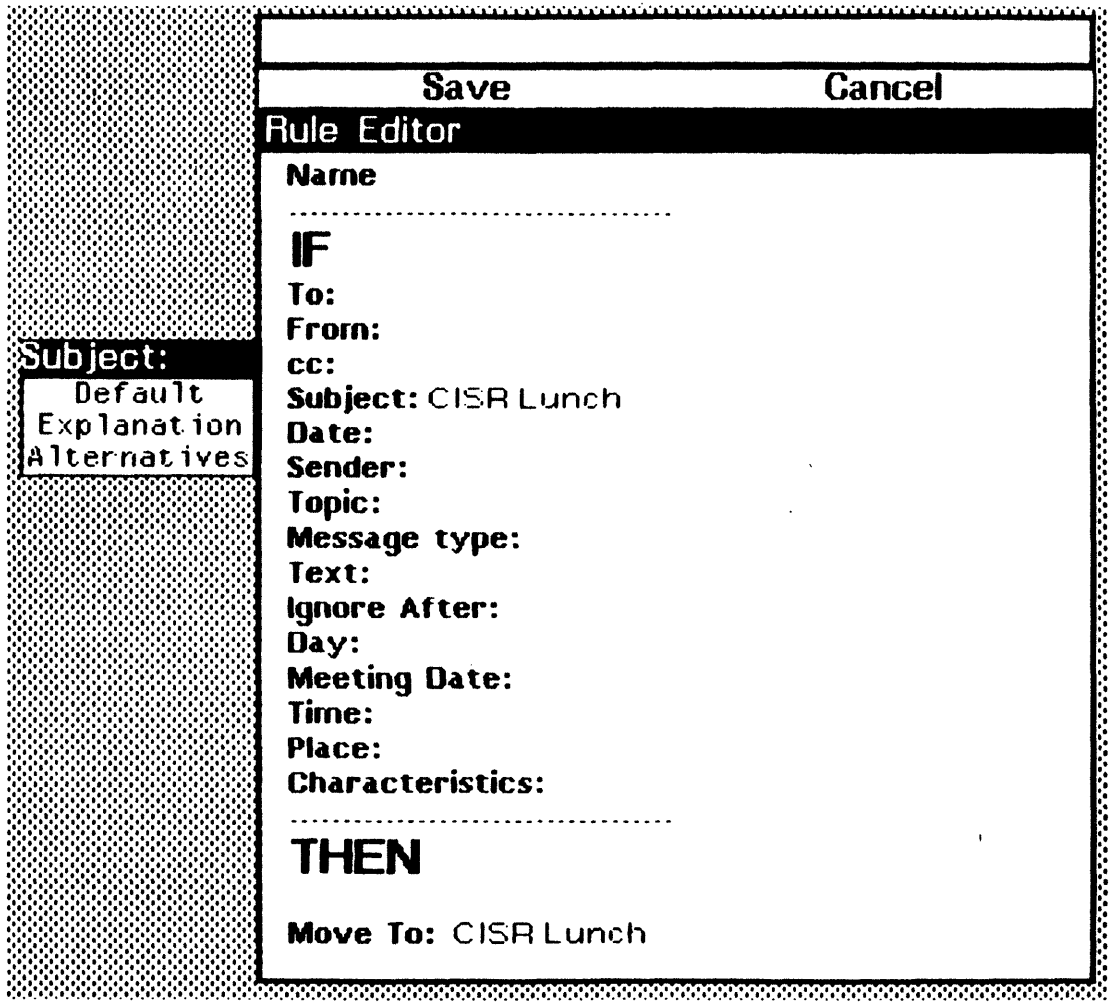

Figure 3

Rules for processing messages are composed using the same kind of editor and the same templates as those used for composing messages in the first place.

Figure 4

Sample rules

(a) **IF** Message type: Action request
Action deadline: Today, Tomorrow
THEN *Move* to: Urgent

(b) **IF** Message type: Meeting announcement
Day: Not Tuesday
THEN *Delete*

(c) **IF** Message type: Request for information
Subject: AI, Lisp
THEN *Show*

(d) **IF** From: Silk, Siegel
THEN *Set Characteristic:* VIP

IF Message type: Action request
Characteristics: VIP
THEN *Move to* : Urgent

(e) **IF** Message type: NYT Article
Subject: Computer
THEN *Move to*: Computers

IF Message type: NYT Article
Subject: Movies
THEN *Move to:* Movies

IF Message type: NYT Article
Article date: < Today
Characteristics: Not MOVED
THEN *Delete*

IF Message type: NYT Article
Characteristics: Not MOVED and Not DELETED
THEN *Move to:* NYT Articles

Intelligent suggestions for responding to messages

The presence of recognizable types of semi-structured messages also simplifies the task of having the system intelligently present options for what a user might want to do after seeing a message. Almost all electronic mail systems provide standard actions (such as "answer" and "forward") that can be taken after seeing a message. We have recently generalized this capability in two ways.

Suggested reply types. First, since either answering or forwarding a message creates a new message, our system suggests options for the type of new message to be created. For instance, when a user selects the "answer" option for a "bug fix request", a pop-up menu appears with three choices: "bug fix commitment", "request for information", and "other". Selecting commitment results in a new "commitment" message being constructed to answer the message. Figure 5 shows a selection of message types and their default reply types.

Suggested response actions. Even more important than suggesting reply types, our system is also able to suggest other actions a user might want to take after seeing a message of a given type. For example, when a user reads a message of type "meeting announcement" another option, "add to calendar" is automatically presented, in addition to the standard options like "answer" and "forward". If the user selects this option, the information already present in the message in structured form (e.g., date, time, and meeting topic) are used to add an appropriate entry to the user's on-line calendar.[1] As another example, when a user reads a message of type "software release" (or any of its subtypes such as "bug fix announcement"), an option called "load file" is automatically presented, and if this option is chosen, the file specified in the message is automatically loaded into the user's system.

Inheritance of message type characteristics

The construction and use of message templates can be greatly simplified if the templates are arranged in a network with some templates designated as subtypes of others. Then the subtypes of a given template can automatically *inherit* from the *parent* template the field names and other properties (such as defaults and alternatives for field values, rules for processing incoming messages, suggested reply types, and suggested response actions). Any subtype may also, in turn, add new fields or override any of the property values inherited from the parent (e.g., see Fikes & Kehler, 1985). For example, Figure 6 shows the simple network of message types in use in our prototype system. The

Figure 5

Sample of message types automatically suggested as replies

Original message	*Suggested reply types*
Message	Message
Action request	Commitment, Request for information, Action request, Message
Notice	Request for information, Action request, Message
Bug fix request	Bug fix commitment, Bug fix announcement, Request for information
Meeting proposal	Meeting proposal, Meeting acceptance
Meeting announcement	Request for information

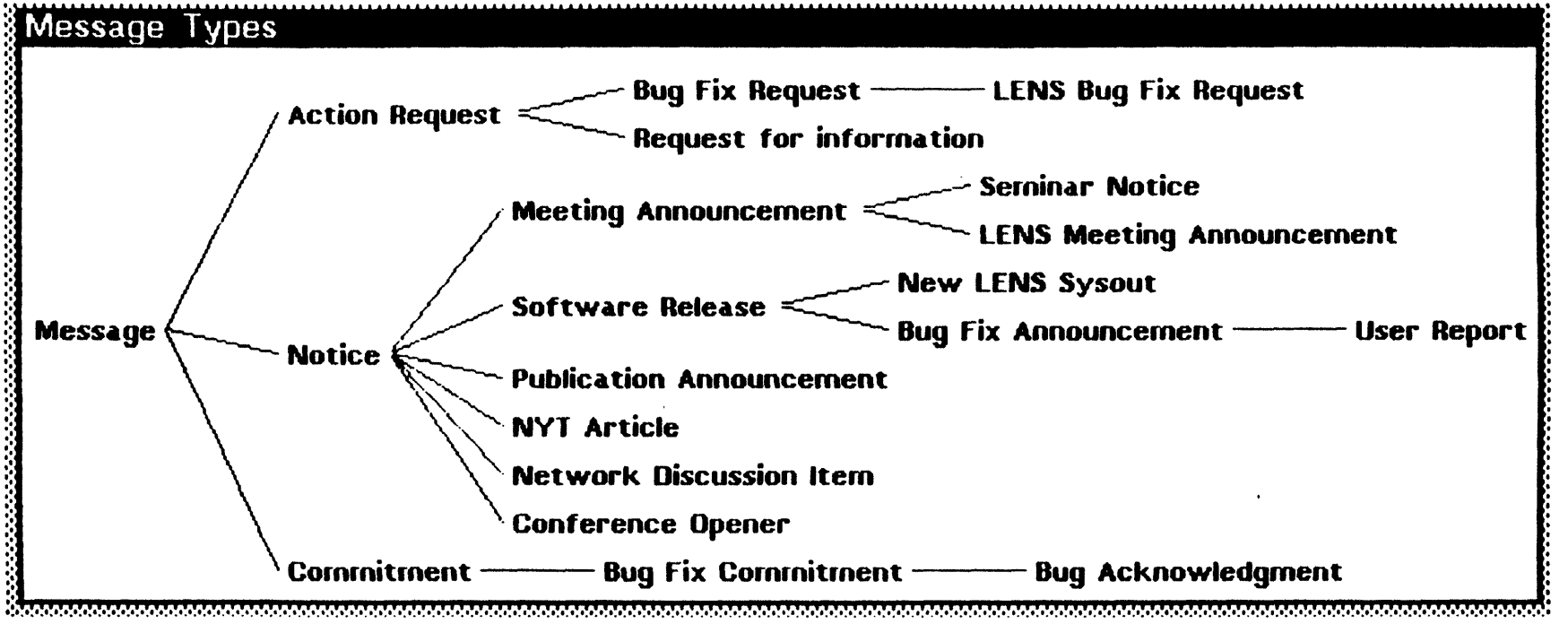

Figure 6

The message templates are arranged in a network with more general types at the "top" (shown at the left) and more specific types at the "bottom" (shown at the right).

seminar announcement template adds a field for *speaker* that is not present in its parent template *meeting announcement*, and the *LENS project meeting announcement* (Figure 2) adds a number of default values that are not present in its parent. The inheritance network eliminates the need to continually re-enter redundant information when adding new templates that resemble old ones, and it provides a natural way of organizing templates, thus making it easier for senders to select the right template.

Applications

Our first, and most fully developed, application of these ideas is the intelligent information sharing system described above. In order to demonstrate the generality of the ideas, we will now describe several other simple applications we have implemented in the same framework. The most important point here is not that it is possible to implement these other applications--in fact, we have chosen applications that have already been implemented in other systems. Instead the point is that a wide variety of applications for supporting cooperative work can all be implemented in a way that (1) is smoothly integrated from the user's point of view, and (2) is much easier to implement because it takes advantage of the basic capabilities for supporting structured messages.

All the applications we will describe are written in the Interlisp-D programming environment for Xerox 1108 series workstations using Loops, an object-oriented extension of Lisp. In order to provide a natural integration of these applications with the capabilities that people already use, our system is built on top of the existing electronic mail system (called Lafite). Users can continue to send and receive their mail as usual, including using centrally maintained distribution lists and manually classifying messages into folders. As of this writing, the information sharing application has been in regular use by about five members of our research group for almost a year and a larger scale test in a corporate research laboratory is about to begin.

Computer conferencing

Many of the capabilities of a computer conferencing system (e.g., Hiltz & Turoff, 1978), can be easily incorporated in the Information Lens system. We have recently done this by (1) adding a new message type called "conference opener" that includes the name of the conference and (optionally) its parent conference, and (2) adding a response option called "join" for this type of message that, if chosen, will automatically (a) create a new folder by this name, (b) create a new rule to select messages addressed to "Anyone" that contain this name in the topic field, and (c) create a new rule to move all messages received about this topic to the new folder.

It is easy to see how our system allows users to supplement these customary capabilities of a computer conferencing system with, for example, more sophisticated rules that filter not only on topic but also on other characteristics such as sender. The primary computer conferencing capability not included in our system so far is the ability for a user to retrieve messages that were sent before that user joined the conference. Here, too, however, the structuring of messages will make possible much more sophisticated retrieval possibilities within the same general framework (e.g., see Tou, Williams, Fikes, Henderson, & Malone, 1982).

Calendar management

We have already seen how a very simple form of calendar management is included in the system by providing users with a response option that will automatically insert incoming meeting announcements into the user's on-line calendar. We have also implemented a more sophisticated protocol for semi-automated meeting scheduling. This protocol uses several new message types, including "meeting proposals" and "meeting acceptances". Proposals and acceptances both include all the fields that meeting announcements do, but the values may often be non-specific (e.g., "sometime this week" for the date field). People can schedule meetings by sending a sequence of proposals (and possibly counter-proposals) until a proposal is accepted.

Our system provides automated support for this process in several ways. First, some people may want to automatically forward all messages of certain types (e.g., meeting proposals) to other people (e.g., their secretaries) for a response. Second, the system helps people construct replies to these messages. For instance, users can choose to reply to a "meeting proposal" with a "meeting acceptance" and all the information (such as time, place, and topic) will be automatically copied from the proposal to the acceptance. When a meeting acceptance is received, one of the action options presented is to "confirm" the meeting. Selecting this option automatically adds the meeting to the user's calendar and sends a meeting announcement to the other participant(s) confirming the scheduled time.

Though we have not done so yet, this general framework also makes it possible to have some meetings scheduled completely automatically (e.g., see Greif, 1982). For example, meeting proposals from certain people (e.g., members of one's own work group) might be automatically accepted if they fall within regular working hours and do not conflict with other meetings already scheduled. Any messages that are exceptional (e.g., a request to meet outside of regular working hours or a request to meet at a time when a conflicting meeting is already scheduled) will then be brought to the attention of the human user for special handling. This property--that exceptional cases are dealt with by

human users in a smoothly integrated way--might be called the "graceful degradation" property. We believe this property will be especially important to the acceptance of systems--like those for calendar management--that involve subtle interpersonal and political issues about which people are reluctant or unable to be explicit.

Project management and task tracking

Systems based on structured messages can support project management and other coordination processes by helping to keep track of what tasks have been assigned to whom (e.g., see Winograd & Flores, 1986; Sluizer & Cashman, 1985; Sathi, Fox, & Greenberg, 1985). One simple way for users to do this in the current system is simply to set up rules that move copies of all action requests and commitments into special folders. For example, action requests a user receives might be categorized in folders by the project to which they relate, while action requests a user sends might be categorized by the people to whom they are sent.

To illustrate how more elaborate capabilities can be built up within the same framework, we have implemented a simple task tracking system for software maintenance activities similar to the example described by Sluizer and Cashman (1985). In this application, "users" of a software system send "problem reports" to a "work assigner." The work assigner first sends an "acknowledgment" to the user and then sends the problem report to a "developer." When the developer fixes the problem, the developer sends the "fix report" to the work assigner who, in turn, sends a "user report" to the original user noting that the problem has been fixed.

This application was implemented in our system by defining the three message types (two of which already existed under different names) and adding several new response options. Users who play the role of work assigners have two possible response actions suggested when they view problem reports. If the problem report has not been acknowledged, the option presented is to acknowledge it. Selecting this option prepares a standard acknowledgment message with the appropriate fields filled in from the information on the original problem report. When the acknowledgment message is sent, the token "acknowledged" is added to the "characteristics" field in the problem report message. Whenever a problem report that includes "acknowledged" in its "characteristics" field is displayed, a response action of "assign" is suggested. Selecting this option prepares the problem report message for forwarding to a developer whose name the work assigner selects from a list of alternatives.

Clearly this task-tracking system is quite limited. The original XCP system, for example, used a set of primitive actions to define protocols involving a number of roles, message types, and actions. A

similar capability would be desirable in our system. Another obviously desirable capability would be more elaborate database facilities for sorting, displaying, and modifying the status of tasks. For example, the current system can sort messages into folders according to various criteria and it can display the header information (e.g., date, sender, and subject) for the messages in a folder. But it would also be quite useful to be able to display the tasks one had committed to do in a report format, sorted by due date, that summarized the task names and the task requestors. If these database capabilities are implemented in a general way for structured objects, then they should be useful for developing many other coordination-supporting applications as well.

Defining the network of message types

Because of the prominent role we are advocating for structured message types, the definition of these types becomes especially important. The network shown in Figure 4 includes some message types that we believe will be useful in almost all organizations (e.g., meeting announcements) and some that are important only in our environment (e.g., LENS project meeting announcement). Different groups and different applications can develop detailed structures to represent the information of specific concern to them. For example, a product design team might have an elaborate network of message types describing different aspects of the product (*e.g.*, market size estimates, response time estimates, alternative power supply vendors).

One of the attractive features of this approach is that it is easy to incrementally increase both the number of message types defined and the number of people who use these message types. For example, individuals who begin using the information sharing system before most other people do can get some immediate benefit from constructing rules using only the fields present in all messages (To, From, Subject, Date). Groups of individuals who begin to use a set of common message types can get much greater benefits from constructing more sophisticated rules for dealing with more specialized message types. For example, individuals in a work group may initially use unstructured electronic mail for scheduling many of their meetings. As this use of the system is recognized as a common and important one, special message types (such as meeting announcements and proposals) might be defined to facilitate sorting and prioritizing these messages appropriately. Eventually, automatic capabilities for scheduling certain kinds of meetings might be added as well. From the viewpoint of organization theory, we know that "internal codes" are among the most important productive assets of an organization (March & Simon, 1958; Arrow, 1974). In effect, the Lens system provides a medium in which this collective language of an organization can be defined and redefined.

Principles for defining message types. In our current system, we have used three principles for defining message types. First, we have based our highest level classification of messages on the *purpose* of the message. The taxonomies of *speech acts* derived by linguists (e.g., Searle, 1975) appear to be quite useful for making these distinctions (see Kedzierski, 1982; Winograd & Flores, 1986). For example, messages whose purpose is to *request information* should be routed to people who know about the topic of the message, while messages whose purpose is to *provide information* should be routed to people who are interested in the topic of the message. Distinctions at this level of generality appear to be quite useful for many kinds of more specific messages. We have also used two more pragmatic principles: (1) Add a specialized message type when a new field would be useful (e.g., the speaker field in "seminar announcements") ; and (2) Add a specialized message type when a new set of defaults would be useful (e.g., the regular weekly time and place in "Lens meeting announcements").

We do not believe that these principles are either necessary or sufficient as a basis for defining message types. We also do not know whether the most useful number of message types will turn out to be relatively small (e.g., a few dozen) or large (e.g., hundreds or thousands). Presumably the answers to these questions depend on the community using the system and the application for which it is being used. One of the important research directions we intend to pursue involves observing how communities evolve a useful set of message types and developing principles (and possibly software) to aid in such a process. For example, we are beginning to explore how the number of message types might be substantially reduced by developing representations for other types of objects that can be included in messages. A single message type called "announcement", for instance, might be used to convey descriptions of many different types of objects: meetings, publications, projects, and so forth.

Message type editor. We have already developed a display-oriented editor, like the message and rule editors shown above, for creating and modifying the message type definitions themselves. As shown in Figure 7, this editor allows people to create new message types, change the fields and field names of existing message types, and change the properties (such as defaults and alternatives) of existing fields. We expect that in some (e.g., rarely used) regions of the network anyone should be able to use this "template editor" to modify an existing message type or define a new one, while in other regions, only specifically designated people should have access to this capability. In the current version of the system, anyone can use a simple version of this editor to personalize the *default*, *explanation*, and *alternatives* properties of the fields in existing message types, and anyone can suggest changes in the

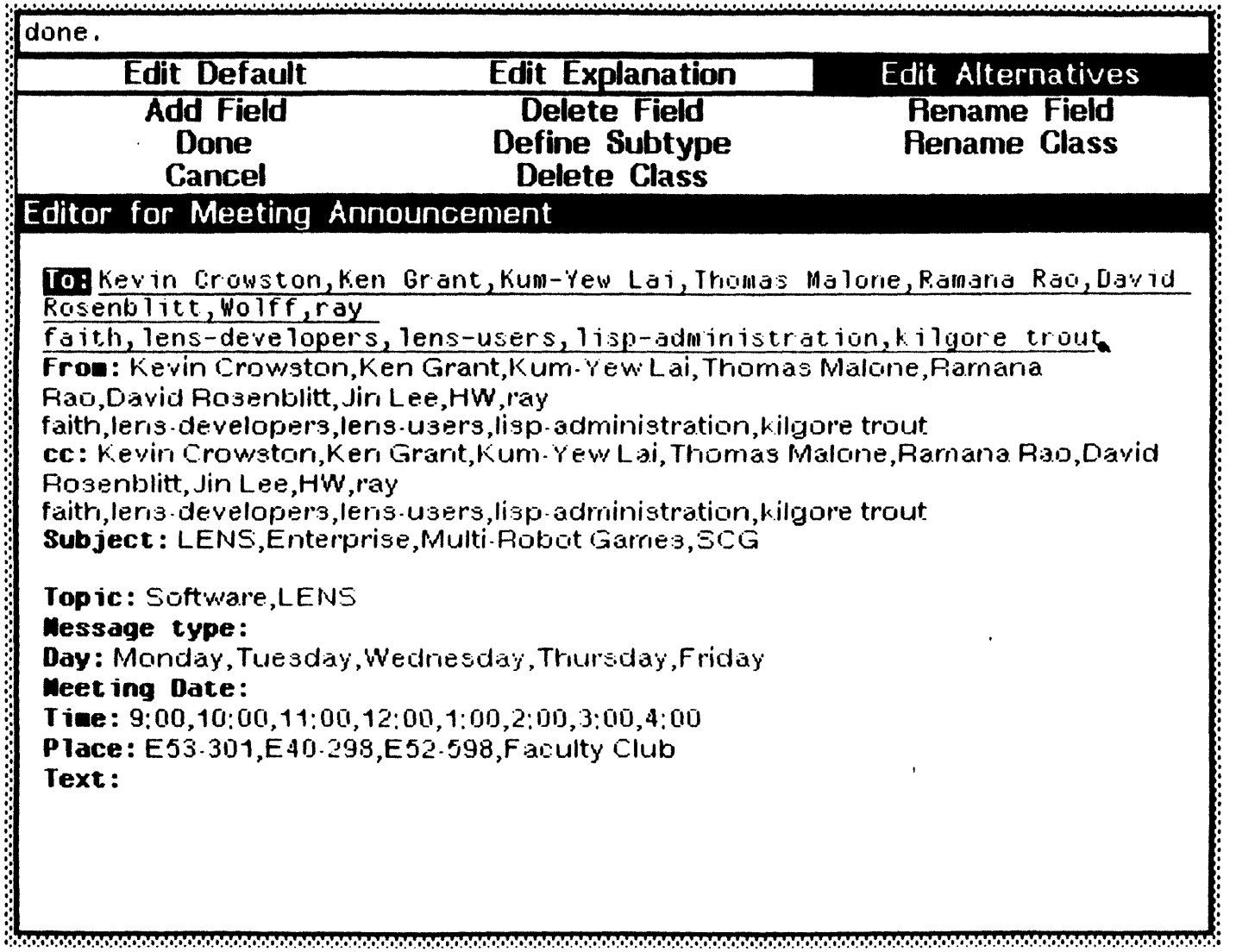

Figure 7

Another graphical editor is used to edit the definitions of message types.

group message definitions, but only the "network administrator" can change the definitions shared by the whole group.

Conclusion

In this paper, we have seen how the use of semi-structured messages can simplify designing systems that (1) help people formulate information they wish to communicate, (2) automatically select, classify, and prioritize information people receive, (3) automatically respond to certain kinds of information, and (4) suggest actions people may wish to take on receiving certain other kinds of information. We believe that systems like this illustrate important--and not yet widely recognized--possibilities for collaboration between people and their machines.

As Stefik (1986) has noted, "the most widely understood goal of artificial intelligence is to understand and build autonomous, intelligent, thinking machines." Recently, however, there have been a number of suggestions of other goals for this field. Stefik (1986), for example, suggests the goal of building an "interactive knowledge medium"; Winograd and Flores (1986) suggest possibilities for designing "tools for conversation"; Luconi, Malone, and Scott Morton (1986) advocate designing "expert support systems" as well as "expert systems"; and Waters (1986) notes the importance of the "assistant" approach to building useful artificial intelligence systems.

We have seen in this paper how a combination of ideas from artificial intelligence and user interface design can provide the basis for powerful computer-based communication and coordination systems. We believe that the power of this approach is partly due to the fact that it does not emphasize building intelligent, autonomous computers, but instead focuses on using computers to gradually support more and more of the knowledge and processing involved when humans work together.

Footnote

[1] The on-line calendar program we use was written by Michel Denber and is part of the user-contributed library of the Interlisp-D system.

Acknowledgments

This research was supported by Xerox Corporation, Wang Laboratories, Inc., Bankers Trust Co., the Management in the 1990's Program at the Sloan School of Management, MIT, and the Center for Information Systems Research, MIT. Portions of this paper appeared previously in Malone, Grant, & Turbak (1986), and Malone et al (in press).

We would like to thank Kevin Crowston, Jin Lee, and Irene Greif for helpful comments.

References

Arrow, K. *Limits of Organization.* Norton, New York, 1974.

Denning, P. Eectronic Junk. *Communications of the ACM 23,* 3, (1982), 163-165.

Ellis, C. & Bernal, M. OFFICETALK-D: An experimental office information system. *Proceedings of the ACM Conference on Office Information Systems,* 1982.

Englebart, D. C. and English, W. K. Research center for augmenting human intellect. *Proceedings of Fall Joint Computing Congress,* AFIPS Press, December 1968, pp. 395-410.

Fikes, R. and Kehler, T. The role of frame-based representation in reasoning. *Communications of the ACM* 28, 7 (1985), 904.

Fox, M., Greenberg, M., Sathi, A., Mattis, J., & Rychener, M. *Callisto: An Intelligent Project Management System.* Unpublished technical report, Intelligent Systems Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, November, 1983.

Gifford, D. K., Baldwin, R. W., Berlin, S. T., & Lucassen, J. T. An architecture for large scale information systems. *Proceedings of the 10th ACM Symposium on Operating Systems Principles* (Orcas Island, Washington, December 1985), pp. 161-170.

Goldstein, I. P. PIE: A network-based personal information environment. *Proceedings of the Office Semantics Workshop,* Chatham, MA, June, 1980.

Greif, I. *Cooperative office work, teleconferencing, and calendar management: A collection of papers.* Unpublished technical memo, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, May, 1982.

Hiltz, S. R. and Turoff, M. *The network nation: Human communication via computer.* Addison-Wesley, Reading, Mass., 1978.

Hiltz, S. R. and Turoff, M. Structuring computer-mediated communication systems to avoid information overload. *Communications of the ACM* 28, 7 (1985), 680-689.

Kedzierski, B. Communication and management support in system development environments. *Proceedings of the Conference on Human Factors in Computer Systems* (Gaithersburg, MD, March 15-17, 1982).

Luconi, F. Malone, T. W., Scott Morton, M. S. Expert systems: The next challenge for managers. *Sloan Management Review,* Summer 1986, 27, 3-14.

Malone, T. W., Grant, K. R., & Turbak, F. A. The Information Lens: An intelligent system for information sharing in organizations. *Proceedings of the CHI '86 Conference on Human Factors in Computing Systems* (Sponsored by ACM/SIGCHI) (Boston, Mass., April 1986).

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. Intelligent information sharing systems. *Communications of the ACM*, in press.

March, J. G. & Simon, H. A. *Organizations*. Wiley, New York, 1958.

Searle, J. A Taxonomy of Illocutionary Acts. In K. Gunderson (Ed.), *Minnesota Studies in the Philosophy of Language*. University of Minnesota Press, Minnesota, 1975.

Sathi, A., Fox, M.S. and Greenberg, M. Representation of activity knowledge for project management. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-7, No. 5 (1985), 531-552.

Sluizer, S. and Cashman, P. XCP: An experimental tool for managing cooperative activity. *Proceedings of the ACM Computer Science Conference*, New Orleans, LA, March 1985.

Stefik, M. The next knowledge medium. *The AI Magazine*, Spring, 1986, 34-46.

Tou, F. N., Williams, M. D., Fikes, R. E., Henderson, D. A., & Malone, T. W. RABBIT: An intelligent database assistant. *Proceedings of the National Conference of the American Association for Artificial Intelligence* (P ittsburgh, Pennsylvania, August 18-20, 1982).

Trigg, R. H., Suchman, L. A., & Halasz, F. G. Supporting collaboration in NoteCards. *Proceedings of the Conference on Computer-Supported Cooperative Work,* Austin, Texas, December 3-5, 1986.

Waters, R. C. KBEmacs: Where's the AI? *The AI Magazine*, Spring, 1986, 47-56.

Winograd, T. & Flores, F. , *Understanding Computers and Cognition: A New Foundation for Design.* Ablex Publishing Corp., Norwood, New Jersey, 1986.