Integrating Coordination Support into Automated Information Systems

by

John Jeffrey Cimral

B.S., United States Military Academy
(1975)
M.A., Pepperdine University
(1978)

Submitted in partial fulfillment
of the requirements for the degrees of

Master of Science
and
Electrical Engineer

at the

Massachusetts Institute of Technology

May 1983

Signature of Author .....

Department of Electrical Engineering and Computer Science
May 13, 1983

Certified by .....

Irene Greif, Thesis Supervisor

Accepted by .....

Arthur C. Smith, Chairman, Departmental Committee

Integrating Coordination Support into Automated Information Systems

by

John Jeffrey Cimral

Submitted to the Department of Electrical Engineering
and Computer Science in May, 1983 in partial fulfillment
of the requirements for the Degrees of
Master of Science
and
Electrical Engineer

Abstract

This thesis explores the issues surrounding the design and implementation of coordination support applications. A role based Multi-Person Calendar (MPCAL) is described. MPCAL facilitates structured coordination, information sharing, and delegation of authority. A role knowledge base provides flexible and precise control over the system. Each role defines information access procedures, enforces integrity constraints, and guides the recording of significant coordination events.

The design criteria presented are derived from several disciplines. Linguistic philosophy is used as a basis for describing coordination primitives. Role theory suggests the use of personal and organizational expectations as a coordination control mechanism. The psychology of man-machine systems highlights the need for usable systems.

Thesis Supervisor: Dr. Irene Greif
Title: Principal Research Associate

Acknowledgments

I am thankful for the support I have received during this work. First, I would like to thank Irene Greif for her encouragement and advice over the last two years. She strongly influenced the final decision to build MPCAL and suggested many interesting approaches to the work. I appreciate the constructive feedback she has always given me.

Andrea Aparo and Dan Carnese both deserve special recognition. They always found time to listen to my ideas and help me over the rough spots. Andrea will always have my greatest respect for his breadth of knowledge and willingness to try new ideas. Dan has taught me to check my assumptions and then check them again.

Several people have given me important advice. I must acknowledge the interest and concern of Marvin Sirbu. Sunil Sarin has helped me since the day I arrived in the group. Dick McKinnon made me appreciate "So What?". Tom Lee made me think.

My friends at M.I.T. are very special. Bob Iannucci, Juliet Sutherland, Stan Zdonik, Renata Sorkin, Nancye Mims, and "the Kids" put up with my craziness. Ann Finn was not only a good friend, but she broke my writer's block by being honest.

I must also thank the people of the U.S. Army and the United States Military Academy for sending me to M.I.T. Your confidence has kept me going.

Most of all my family has made this possible. My wife, Joyce, has supported and encouraged me every day. She is the light of my life. My daughters, Heather and Melanie, are very special. They always wait for me to come home and I hope they will not have to wait so long again. Yes girls, it is only a *'Puter*. Finally, I must thank Kay Duffy for teaching me how important it is to relax and learn from everyone you meet.

Table of Contents

Table of Figures

Chapter One

Introduction

The quality of office work can be improved by computer systems. Traditionally office automation has emphasized applications that improve the efficiency of individual work. In contrast, this thesis is dedicated to the design and implementation of coordination support applications. These systems can facilitate group activities, improve the distribution of work requirements, and track the progress of projects. Coordination based systems "add value" to office work.

Automated coordination support facilities are not available today. In the past, economic and technical barriers stood in the way. These barriers have been reduced, but new roadblocks lie ahead. Some of these concerns are:

- Whether or not an application can draw upon a general conceptual model of coordination.

- The fundamental nature of computer communication raises new issues in interpersonal coordination. Functional extensions to basic conceptual models are necessary.

- Coordination systems must recognize individual capabilities.

- An application for coordination support must incorporate organizational and individual behavioral expectations into its design.

- The system designer cannot ignore basic "usability" issues.

This thesis has tried to address each of these concerns with practical insight and

theoretical discussion.

The M.I.T. Office automation group has built several automated calendar systems as research vehicles for developing computer based coordination support mechanisms. Even though previous calendar projects were fruitful, an effort to consolidate and extend the application theory was necessary. The Multi-Person Calendar (MPCAL) took the work of PCAL, OPTICAL, SHARACAL and 201CAL, revised the basic coordination model, and extended information sharing and delegation of authority capabilities with an independent role based control mechanism.

MPCAL is used to highlight the design and implementation requirements of a computer based coordination system. It supports calendar browsing, information sharing, structured meeting conversations, and delegation of authority. It can be tailored for various calendar applications, and it is capable of tracking significant calendar events. MPCAL also features an integrated interface design. The display structure, command language, on-line assistance and feedback messages have all received special attention.

Coordination on a computer system requires more then a structured conversation model. Information sharing and delegation of authority are important parts of an automated coordination system. Information sharing reduces the need for expensive and time consuming secondary conversations. Delegation of authority

allows one person to legitimately act for another. This latter facility can be used to filter calendar requests, redistribute work load and increase the probability of important information being seen.

An effective coordination support mechanism requires information about the people using it. Capturing individual and group behavioral expectations is vital to the system's success. MPCAL has taken role theory from the social sciences and applied it to calendar design. A role in MPCAL captures organizational and individual behavior expectations for people using a calendar. Each role defines information access procedures, enforces integrity constraints and records significant calendar events.

MPCAL has a separate role knowledge base. Each role is a set of rules that encapsulates organizational and individual expectations. MPCAL can be tailored to the organization it serves. Predefined system roles allow the organization to specify general expectations for individual capabilities. Each person also has the ability to define new roles to meet their needs. The separate role base permits tailoring. It is an effective control mechanism separate from the remainder of the system. This separation allows easy inspection and modification of the control structure.

"Usability" is a critical concern for every system designer. It goes well beyond the user interface design. It is easy to lose track of a project's primary goal unless the audience's needs are consciously and continuously integrated into every design

decision. User expectations require careful consideration. Physical and conceptual requirements are important. An appreciation for user anxiety during initial encounters with the system must be cultivated. "Usability" is the bottom line.

1.1 Thesis Overview

The remainder of this thesis is divided into four chapters.

Chapter Two describes the M.I.T. Office Automation Group's interest in coordination support. ECOLE, an advanced office workstation project, is outlined and the methodologies used to develop ECOLE software modules are summarized. Earlier group work in calendar coordination is presented. Chapter Two sets the stage for the Multi-Person Calendar Project (MPCAL).

Chapter Three presents the design, implementation and evaluation of MPCAL. MPCAL's facilities for browsing, coordinating meetings, supporting different calendar types and recording significant events are discussed. MPCAL's interface design is reviewed. The implementation is described with special emphasis on the program's structure and unique role database. MPCAL is then critiqued in terms of lessons learned and areas for improvement.

Chapter Four presents the justification for MPCAL's design. A coordination model is presented and several meeting conversations are analyzed. Information sharing and delegation of authority issues are reviewed. Role theory is described as a

basis for the MPCAL role facility. This chapter also presents four criteria for building usable systems. These criteria emphasize a general perspective on system design that transcends more traditional concerns for user interface design.

Finally, Chapter Five summarizes the work and consolidates the key findings. Coordination support applications and roles are briefly critiqued. Recommendations are made for further research.

Chapter Two

Background

This chapter summarizes the M.I.T. Office Automation Group's interest in coordination support. It discusses the relationship between coordination support and the design of an advanced office workstation, ECOLE. The evolution of the Personal Calendar (PCAL) application, as a coordination support test bed, is described. Finally, the stage is set for the design and implementation of the Multi-Person Calendar (MPCAL).

2.1 Office Workstation Design

Coordination support research in the Office Automation Group has always taken place within the larger context of ECOLE, the advanced office workstation project. ECOLE is dedicated to the development of an office application builder's tool kit. Each individual tool in ECOLE serves a particular function. In combination, the tools can effectively fashion a collection of integrated office applications, that share a common interface and are easy to maintain.

Two methods of tool design and implementation have been used in ECOLE. They are:

1. Application Based Design: An application is designed and built.

Subsequently, generic tools are constructed based on this application experience.

2. Conceptually Based Design: A conceptual requirement for the system is formulated. A generic tool is built to meet the requirement. The tool is tested in an application.

Both approaches have been successfully used to develop ECOLE software tools.

The Easy To Use Document Editor (ETUDE) [28] is an example of application based tool development. ETUDE was designed for ease of use and ease of learning. It is a "what you see is what you get" text editing system. During the process of building ETUDE, the ECOLE command parsing and window management packages were implemented. As ETUDE was gradually developed, the underlying packages were continuously refined and expanded. When ETUDE's interface was tested [21], user feedback was not only integrated into the editor, but also contributed to implementation of the underlying ECOLE packages. The entire process of designing, implementing and testing an application provided important insights at several design levels.

The Extensible and Natural Common Object Resource (ENCORE) exemplifies the second method of tool design. It is an "object management system" intended to replace more traditional file and database systems [52]. ENCORE benefited from the ETUDE project. It was greatly influenced by the ETUDE experience. In contrast to ETUDE, ENCORE is an ECOLE package being constructed before it's integration into an application. ENCORE is designed to meet the unique data

storage requirements of the office environment. It can maintain several object versions to support multiple views. It can also establish complex relationships between objects and maintain these relationships automatically. When ENCORE is complete it will be tested in a joint document writing application.

2.2 Office Calendar Coordination

Coordinated activities are a critical part of every office. ECOLE would be incomplete without tools for building coordination support applications. Constructing these tools requires a clear understanding of the office coordination process and its support requirements. As a research aid, calendar applications have been used to explore office coordination.

There are two primary goals for the group's calendar research efforts [25]. They are:

1. To provide software support for groups of people working together, whether within one organization or across organizational boundaries;

2. To provide software development tools and design guidelines for building systems that will be used by groups of people, in evolving work situations.

Modeling how people coordinate work in an organization illustrates what coordination support is required. In a group activity, each person adopts a set of appropriate behaviors based upon relative status, the task at hand, and the accepted norms of the group [42]. People coordinate and communicate according to the role

they choose, or are assigned by the people around them. Working relationships develop between individuals and work is completed in a coordinated manner. Therefore, to effectively support a group, an automated system must support the coordination and communication protocols that exist within the group.

Several experimental calendar systems in the Office Automation group have sought to develop coordination features based upon individual roles. It is important to review this work, since it forms the basis for this thesis.

2.2.1 The Personal Calendar (PCAL)

The original PCAL [24] was the first calendar system built in the Office Automation Group. It defined the basic single user calendar functionality that has remained the common heritage of every calendar version. Implicitly, PCAL enforced the rights of a single calendar owner. The owner could create, change, and cancel appointments, but support for meeting coordination was limited. It was possible to "LOOK AT" other calendars, but impossible to change foreign calendars or view several calendars simultaneously. There were no significant features that supported cooperative scheduling, and this was a major drawback.

Consequently, PCAL was expanded to include two elementary coordination features. A secretary "role" was created which could be assigned to other calendar users. The secretary could not only see the calendar, but he also had extended capabilities that permitted him to act for the owner. The secretary was allowed to

create, change, and cancel appointments for the owner without restriction. The second feature added to PCAL was a command that checked the schedules of a meeting's participants for conflicts. The command returned a report on apparent scheduling conflicts and listed participants that did not have calendars. These extensions reflected a growing appreciation for coordination support. The simple mechanization of a common office calendar was not very useful, so these extentions were added to increase PCAL's support for cooperative activities.

PCAL extensions demonstrate some potential types of coordination that are possible on a computer system. The major requirements for information sharing and personal privacy in this "multi-person" system were recognized, but there were many unanswered questions about appropriate functionality, the control of the sharing process, and the proper interface design. Other calendar versions have helped answer these questions.

2.2.2 The Option Calendar (OPTICAL)

OPTICAL, like all subsequent versions of the calendar, is a PCAL derivative. It was the first of several experimental calendars that explored the broader use and definition of roles. In OPTICAL roles were defined as a collection of calendar access rights that could be assigned to an individual. They were used to support information sharing while preserving some degree of privacy.

OPTICAL defined a role as a collection of access rights to various time periods

in a calendar. There were four types of access rights in OPTICAL; full access, read only access, filtered read access, and no access. Each day was also broken down into several time periods, such as office hours or business hours. The time period definitions could be changed. Each appointment in this calendar fell into a defined time period. A role was defined by the type of access it had to each time period. For instance, the owner had "full" access to all time periods, while the secretary might have "full" access to all times except "private hours".

OPTICAL had three built-in roles and it allowed the definition of new roles. The owner, secretary and public roles were standard. A new role was created and assigned to users in several steps. First a unique name was assigned to the new role. Second, each time period was given one type of the access. Finally, a user could be assigned one role that controlled everything he could see and do in a particular calendar.

OPTICAL was a useful exercise for several reasons. It clarified the idea of a user "role". Roles served as a valuable abstraction mechanism for the definition of individual information sharing abilities. Additionally, the names associated with roles carry enough "conceptual baggage" to make them recognizable without having to understanding every part of the role. A well named role has an intuitive feel. For instance, people have behavioral expectations for a secretary. When these expectations can be built into a system, it becomes easier to learn and use.

Default roles were very important in OPTICAL. They provided common points of reference between calendars. If a good default role set is available, most casual users never define new roles. In OPTICAL's case the roles were useful, but more could have been predefined.

OPTICAL also demonstrated no matter how many default roles are defined, a facility for defining new roles is necessary. Since defaults are fixed, they cannot meet the expectations or needs of every user. A casual system user should be able to define new roles effectively.

OPTICAL highlighted the importance of appointment classification. Using the combination of time period classifications and access rights was unwieldy. The relationships between a day's time periods attributes and access rights were too obscure. It was always difficult to determine why a command was possible during one period and not during another. Classification of appointments for access became the next topic of calendar research.

2.2.3 The Shared Calendar (SHARACAL)

After the work on OPTICAL two other calendar projects were started in parallel. SHARACAL [32] was one of these. SHARACAL tried a new approach to appointment classification. It also explored the problems of maintaining several versions of a shared calendar.

Explicit classification of appointments is one means for supporting information sharing and maintaining privacy. SHARACAL took this approach. Whenever an appointment was created, it could be classified explicitly as either personal or private. The three roles, owner, secretary and public were built on these classifications. The owner had full access to every appointment in the calendar. A secretary could not create or see the details of personal appointments and the public could only see and create unclassified appointments. If a person did not have access to a particular appointment type, only the appointment's outline was shown in the various calendar displays.

SHARACAL was the first calendar version to recognize the importance of highlighting calendar changes. When a calendar is shared, some users are very interested in being able to identify changes that other people make. For example, if a secretary makes an appointment for a manager, the manager probably wants the new appointment highlighted to make sure the new commitment is seen. SHARACAL did this by flagging appointments in different roles. SHARACAL maintained five types of flags for new, unseen,changed, canceled and action appointments.

SHARACAL represented an important milestone. Roles effectively supported the calendars functionality and the highlighting flags helped organize the interface.

SHARACAL also was not perfect. It had major functional and conceptual

defects. There was no way to create a new role, and role access was based upon the single appointment classification attribute. Explicitly classifying appointments made the user's job more difficult. The classifications were one more thing a person had to understand before the system was usable. Another means of classification is required.

2.2.4 The Room Calendar (201CAL)

201CAL [36] was another experimental calendar that explored the role concept. It defined more roles based upon two general role types. A calendar was classified as either a resource or personal calendar. Personal calendars were the same as calendars in PCAL, and resource calendars were used to allocate limited resources such as conference rooms. The fixed resource calendar roles were:

- Owner
- Requester
- Allocator

An owner was responsible for the room, but did not manage it. A requester could reserve a resource on a first come, first served basis. A requester could only modify appointments that he made. On the other hand, an allocator could change or cancel any appointment and was able to schedule conflicts. This role refereed the resource. An especially interesting feature of the 201CAL was the ability to "ASSUME" other roles. By "changing hats", a person could narrow or expand his

calendar view based on the set of roles he had been assigned.

2.3 Why Another Calendar?

Despite the limitations of previous versions, significant lessons have been learned. Role definitions can be used effectively to control the sharing of information and limit the ability of outsiders to act on personal calendars. Predefined roles, with natural defaults, are useful for new users. Experienced users not only want to modify existing roles, but also want to create entirely new roles. The process of creating a role should be separate from the action of assigning users to the role. Finally, it should be possible for a user to move freely between several assigned roles.

All of the previous versions of the calendar have run into problems in one way or another. Each was completely dependent on some "coded in" feature of the implementation.

- PCAL, SHARACAL, and 201CAL roles were fixed and could not be modified. Neither customization nor extensibility were allowed.

- OPTICAL allowed the definition of new roles, but only based on fixed time period classifications.

- SHARACAL attacked the classification problem by defining fixed roles based on access to fixed appointment types.

- Either OPTICAL or a logical extension of SHARACAL would place a heavy burden on the appointment creator.

Difficulties with extending role definitions and classifying appointments has limited the effectiveness of all previous calendar implementations.

The Multi-person Calendar (MPCAL) Project has addressed these difficulties.

Chapter Three

MPCAL

This chapter describes the Multi-Person Calendar (MPCAL) research project. The design and implementation of MPCAL are presented, followed by an evaluation of the project. The project's primary goal is to identify the design and implementation requirements of a computer supported office coordination system. It explores the type of coordination support that is possible when basic functionality, information sharing, delegation of authority, and effective user interface design are integrated into a single application.

It is useful to make some observations about MPCAL before discussing its general functions. MPCAL does not replace more traditional means of appointment scheduling. It does complement them. Face to face conversations, mail and phone calls are commonly used to schedule meetings. MPCAL is useful when the other means are impractical or inconvenient. It offers a structured environment for calendar management.

MPCAL supports coordination better than previous calendars built by the Office Automation Group at M.I.T. Previous calendar projects focused on supporting several people sharing a single calendar. MPCAL coordinates meeting conversations between calendars. Previous calendars acted as a static recording mechanism.

MPCAL supports dynamic coordination. MPCAL's general appearance is similar to previous calendars (see Appendix A), but its functionality is greatly expanded.

3.1 Design

MPCAL provides a complete set of calendar functions. Much of its functionality is directed toward the support of meeting scheduling. There are commands for exchanging meeting proposals and confirmations. It is possible for people to share calendar information and still maintain control over the sharing policy. MPCAL can be tailored to act as either a personal, group or resource calendar. MPCAL also records significant calendar events, notifies the "principal" calendar user of new commitments and highlights calendar changes that require either confirmation or rejection.

Additionally, MPCAL includes significant interface features that support the functionality. The command language is designed to provide a uniform dialogue. The display layout is carefully structured. Finally, there are extensive help and feedback facilities.

3.1.1 Functionality

MPCAL's primary functional objective is to facilitate meeting scheduling. It does this in two ways.

1. Ad hoc browsing through several calendars is possible either by visiting several calendars sequentially or by checking several calendars in

parallel.

2. A structured conversation facility exists for creating meeting proposals and tracking responses to these proposals.

Both the ad hoc method and the structured method of meeting coordination are effective.

Calendar Browsing

Browsing through calendars in an ad hoc manner is useful when a person is trying to schedule a meeting. There are two separate methods for browsing through MPCAL calendars. First, the "VISIT" command allows a person in one calendar to switch to another. The use of this command may be restricted. The calendar being entered may either deny access or the information displayed may be modified to reflect only blocks of busy time (See Figure A-11). A person might use "VISIT" to manually collect information from several calendars and then return to their own calendar to schedule a meeting.

Instead of sequentially visiting several calendars to gather information, it is possible to "CHECK" several calendars at once. Either "CHECK DAY" or "CHECK WEEK" can be used to collect and combine information for a structured report (See Figure A-12). This command has the same effect as visiting several calendars, but it keeps track of the information for the user. "CHECK" can directly support the scheduling process.

Structured Meeting Conversations

The ad hoc browsing facilities in MPCAL are an extension of the functions available in previous calendar versions. In contrast, the support for "structured meeting conversations" is unique to MPCAL. Even though ad hoc information sharing functions are useful, MPCAL conversations are a more effective and complete method for scheduling meetings.

A *structured meeting conversation* is a clearly defined method for exchanging meeting proposals and responses to the proposals. Each MPCAL conversation follows a general pattern of interaction.

- A meeting proposal is created.
- The proposal is distributed to other calendar users.
- Each calendar user either explicitly responds to the proposal or a rejection is assumed after an RSVP date.
- All responses to a proposals are recorded and available for review by the meeting caller.

Each of these steps is fully supported in MPCAL.

Creating a proposal is the first step in an MPCAL conversation. A proposal is a type of appointment object. These appointments include enough information to make the proposal meaningful (see Figure A-10). For instance, every proposal includes a date and time. Optional information such as an end time, keywords,

participant list and general comments may also be included. The identity of the person creating the object is automatically part of the appointment proposal.

The next step in a conversation is the distribution of the meeting proposal to other people. Distributing a proposal requires the creation of a distribution list, and the attachment of an RSVP date. The distribution list determines which calendars receive a proposal. The RSVP date assures every conversation is completed in one way or another. Failing to reply by the RSVP date is an implicit rejection of a proposal.

There are several ways a person can respond to a proposal. A proposal can be confirmed, rejected or held for later action. If an appointment proposal is confirmed a commitment is made to attend the meeting and the conversation is complete. If the proposal is rejected, the person who made the proposal is notified and the conversation is complete. If a proposal is held for action, the meeting caller may be optionally notified.

MPCAL deals with two problems that do not arise in face to face conversations. First, there is a potential problem with delivery of proposals, commitments and rejections. It is impossible to absolutely guarantee any MPCAL conversation message will arrive at its destination. Second, even if a message is delivered there is no way to insure somebody will see it. For example, a calendar user may either be out of town or may forget to check the calendar for a few days.

In order to deal with these difficulties, MPCAL has adopted a fixed acknowledgment policy based on three rules. These rules are:

1. Every message sent to another calendar is either immediately delivered or never delivered. A person sending a message always knows whether or not it is delivered. Every appointment contains a complete history of both successful and unsuccessful attempts to deliver proposals.

2. Whenever a user looks at the details of a pending proposal, there is an opportunity provided to notify the person who sent the proposal. This second rule addresses the lack of user presence by keeping the proposal creator aware of who has seen the message.

3. All confirmations and rejections, responding to a proposal, are automatically returned to the calendar where the proposal originated.

These three rules try to insure people on the system always know what has happened to proposals they make.

Information Sharing in MPCAL

MPCAL's information sharing facility enhances the process of creating meetings. Information sharing in this application is designed to make initial proposals in a conversation more feasible. A proposal is feasible if it is based on the best information available to the meeting creator about the participant's schedules. In order to create feasible proposals MPCAL users share information about their calendars. This facility reduces the likelihood of expensive conversations about alternate times for the same meeting.

Information sharing in MPCAL respects individual privacy requirements.

Calendar information can be very sensitive and personal. The calendar strikes an equitable balance between sharing and individual privacy. MPCAL only shares a minimal amount of information when it supports the creation of meeting proposals. Each information sharing calendar releases a single day's schedule outline. This information only includes meeting times and whether a meeting is confirmed or not. No keywords, participant lists or other information leaves a calendar. The potential for abusing this limited amount of information is slight. Every person in the calendar can potentially benefit from sharing, since time consuming secondary conversations are less likely.

The fairness of information sharing in MPCAL is another privacy concern. Fairness is judged according to three criteria [17].

1. Each person should be able to decide whether or not they wish to share information.

2. The information should be used only for its intended purpose.

3. Precautions should be taken to insure information is not misused.

MPCAL meets the first criteria by giving each person the capability to specify who may and may not share information. Since information sharing only supports meeting creation the second criteria is met. Finally, disclosure of shared information to third parties is controlled in MPCAL. The calendar outlines are only available to produce the temporary display seen during the appointment creation process. The information is discarded when the process is completed. There is no

way for the limited information to be distributed on the system. Of course, there is nothing that can preclude the information from being copied down while the report is visible, but the limited nature of the shared information reduces the potential for misusing the information.

In MPCAL sharing takes place while a new proposal is being created (see Figure A-13). The date and participant fields of a proposal are used to define what day and which calendars are important. Each listed participant's calendar is checked and if information sharing is allowed an outline of the day is returned. The outlines are merged together and a report is produced that highlights a feasible set of meeting times.

The report is presented in two parts. Each calendar in the participant list is asked to share information. If a calendar does not exist or information sharing is not allowed, the shortcomings are presented in the first part. The report then highlights free periods each calendar has in common.

Information sharing is not required during proposal creation. Gathering calendar outlines takes time. A person may only be interested in a specific meeting time, or another means of communication may have already established the meeting and the verbal commitment is simply being recorded in the calendar. In either case, the report is unnecessary. Whenever a meeting is created without using a form, participant calendars are not checked. When a form is used calendar checking must

be explicitly requested.

Delegation of Authority in MPCAL

MPCAL's facility for delegating authority is a critical part of the system. One principal actor exists in every calendar. Only the principal is committed by proposals or confirmations made in a particular calendar. In addition to the principal, a calendar may have several people who are authorized to take part in conversations. The principal can delegate authority, but never responsibility to other people. A secretary, for instance, may have authorization to schedule meetings, distribute and answer proposals for a manager. In this case, the secretary may act for the manager, even though the manager remains responsible for any action the secretary takes.

Delegation of authority fulfills two major needs. First, it permits action even though the principal is absent. This reduces problems stemming from a lack of principal user presence on the system. Additionally, very busy individuals may authorize other people to act as filters for them. Often an assistant can decide what should be done with a proposal without ever bothering the principal. The principal's time is saved for the most important decisions when other people are authorized to take part in conversations.

This concludes the summary of MPCAL's support for meeting scheduling.

MPCAL provides for ad hoc and structured scheduling. Ad hoc scheduling may use the "VISIT" and "CHECK" commands. Structured scheduling follows a coordinated sequence of actions that include methods for creating, distributing and responding to meeting proposals.

Supporting Different Calendar Types

A calendar in MPCAL can be tailored to serve as a personal, group or resource calendar. This flexibility allows MPCAL to serve several purposes. For example, a personal calendar is used by one person to record meetings and generate meeting proposals. The information in a personal calendar may be protected to insure individual privacy. Information sharing can be limited or the ability for other people to "visit" the calendar can be restricted.

On the other hand, a group calendar can serve as a central information repository for several people. Members of the group might be allowed to schedule and confirm meetings, while group supervisors can reserve the rights to cancel meetings and assign new group members.

A group calendar has been used in the M.I.T. Office Automation Group. It is records group meetings, announces social events, and lists interesting seminars. Each member can "visit" the calendar, add new meetings, or copy appointments to their own personal calendars. The group supervisors may also define group membership,

confirm proposals, and cancel commitments.

An MPCAL calendar can also manage scheduled resources, such as rooms. It is possible to create a MPCAL calendar for a room, assign a person to manage it and define a policy for scheduling its use. For instance, a secretary might manage a room on a first come first served basis. Proposals could be made for times that are not shown as committed and the secretary could decide between conflicting proposals.

Calendars can be used for many things. The primary point is that MPCAL is flexible enough to support many types of calendar activities.

Tracking Significant Calendar Events

MPCAL has special facilities for recording, highlighting, and notifying calendar users about significant events.

Calendar events can have different degrees of importance. MPCAL uses three methods showing significant actions. Their use is situation dependent. For example, an action such as "confirming" a meeting proposal may generate different side effects depending upon who is taking the action.

Some MPCAL objects, such as appointments and reminders, have a history associated with them. Often it is important to record when a meeting is created, changed or seen by some person. Actually any action in MPCAL, that affects an

object with history, may be recorded. The calendar can automatically add a notation about an action, including a date, time, and the actors name. The history is a running commentary on important events. It can be seen with the "SHOW APPOINTMENT" command (See Figure A-3).

Important changes to the calendar can be highlighted or "flagged". New or unseen proposed appointments, changed or canceled committed meetings often require special emphasis at the interface. An appointment can be "flagged" in various displays to emphasize their importance (See Figure A-4).

Since MPCAL allows the delegation of authority, special issues arise. In some situations, it is extremely important to notify a person that a commitment has been made for them. In a system without delegation of authority, the individual who is responsible for a conversation is the only person authorized to make a commitment and principal is always assumed to be aware of his proposals and commitments. This assumption does not hold when other people are authorized to act. When delegation of authority is allowed principals must be notified of proposals and commitments made in their behalf. MPCAL supports notification is several ways.

Notification is more than just highlighting changes in a calendar. New proposals, canceled or changed meetings require highlighting until they are acted upon. Commitments must be seen by the principal on the system. Highlighting changes in the calendar is not as critical as notifying the principal of new

commitments. A proposal, for example, may be ignored and it will be removed after its RSVP date. A commitment can never be ignored because other actions may be dependent upon it.

MPCAL recognizes the notification problem and deals with it in two ways. First, the principal in a calendar always sees new commitments at the beginning of any MPCAL session (See Figure A-5). They are summarized in a short report format. Second, if a commitment date approaches without the principal being notified, the person who authorized the commitment is requested to notify the principal by some other means.

Notification, highlighting and history are used extensively in MPCAL. They may be used separately or they may complement each other. They are an important part of the calendar's functionality.

3.1.2 Features

MPCAL's functionality as a calendar system cannot be separated from the features that make it conceptually compatible with the people who use it. The display layout, command language, help facilities and feedback messages all contribute to the cognitive interface. These features are described in this section.

Displaying MPCAL Information

Each MPCAL display is a carefully designed structure for displaying information. Various screen areas serve different purposes. There is an area for context information about the particular calendar. There is an central display area for presenting different views of calendar objects and help information. At the bottom of the display are the command and feedback lines.

The top two lines in MPCAL display context specific information about a calendar (See Figure A-1). It includes the calendar's name, the current user's name and active role, and the last command. In the case shown in A-1 CIMRAL's calendar is being displayed for JOYCE. JOYCE is acting as a secretary and the last command was "SHOW DAY".

The central portion of the screen is used to display calendar objects, such as appointments, reminders, days or weeks. In Figure A-1, the default MPCAL display, "SHOW DAY" lists one day's appointments and reminders, along with a separate list of highlighted appointments. The command "SHOW WEEK" displays (Figure A-2) appointment proposals, and commitments for an entire week. "SHOW APPOINTMENT" presents the complete details of a particular appointment next to a day summary (See Figure A-3). Notice each presentation presents a different view of the same appointment at 9am Tuesday, 10 May 1983.

The central screen area is also used to display help, command forms and long error messages (See Figures A-8 and A-10). These displays temporarily "pop up"

over the main display.

The bottom lines of the screen are used for the command entry and feedback messages. As a command is entered it appears after the "MPCAL >" prompt. The line above the command line is used to highlight feedback message from the system.

The MPCAL Command Language

MPCAL's command language presents a familiar calendar model. It capitalizes upon a user's prior knowledge and expectations for a calendar by consistently applying a limited set of actions, such as "SHOW" and "CHANGE", to a limited set of objects, such as "WEEK" and "APPOINTMENT". The language is self documenting and task specific.

In MPCAL every command applies one action to one object. Each command's execution time is relatively short. Whenever processing may exceed a couple seconds, the user is reassured with a specific interface message. Finally, the major portion of processing and updating is done when the command is completed. The user is released to start thinking about the next action, even though the machine is still working.

MPCAL uses a consistent set of actions to manipulate every object in the system (Figure A-6). The same action names are always used to create, change, display, and delete objects. For instance, after a person uses the command "CHANGE

APPOINTMENT" then he knows that other object may be modified by "CHANGE". He can find which objects are modified by entering "CHANGE" followed by a "?". A list of MPCAL objects is then displayed (Figure A-7).

MPCAL allows two styles of dialogue. There is a command line interface that may either be used to specify details directly or may expand into a form (Figure A-10) when important information is missing. Novice and casual users often use forms, while experts use direct entry.

The MPCAL Help Facility

MPCAL has an extensive on-line help facility. There are "HELP" and "?" commands that provide general and situation specific information about the use of the system. The general "HELP" facility tells what a function does, its composition, effect, generic type, escape procedures and how to reverse its effect (Figure A-8). The situation dependent facility, "?" gives specific information. It describes current options, and format rules (Figure A-9).

MPCAL Feedback Messages

Feedback messages in MPCAL are used for several purposes.

- They reduce user frustration associated with time delays.
- They inform the user when a action is successfully completed.

- They return specific error information when a command is not completed.

Feedback messages help reduce user frustration in MPCAL. The execution time for MPCAL commands can vary significantly. Some commands, such as "SHOW DAY", are very fast. Other commands, such as "CHECK WEEK", are relatively slow. The amount of processing and communication each command must do is variable. People get frustrated when they finish some part of a dialogue and must wait for the machine. Messages do not eliminate the problem, but at least they reassure the user something is being done.

MPCAL always tells a person when it completes an action. The message confirms the last action took place successfully. This information releases the user to work on other tasks.

MPCAL also provides detailed information when a command fails. There can be any number of reasons for an action being rejected. For instance, assess may not be allowed when a person tries to visit a calendar. Error messages tell the user what has happened and often point to some method for correcting the situation.

3.2 Implementation

MPCAL is a large application program. Even though its appearance is similar to previous calendars, it is structured very differently. It took approximately 1000

hours of work to bring MPCAL up and it is continually being refitted and debugged. The most significant aspects of the implementation are its general structure and its unique role facility. These two parts of MPCAL are reviewed in this section.

3.2.1 MPCAL's Program Structure

MPCAL's code is structured in a straightforward manner. There are three levels of modules. The top level controls the start and finish of a session and it contains the primary command loop. The second level has a module for every MPCAL object. Every action that can be applied to a specific object type is contained in the object module. The lowest level of the code includes the window manager, command parser, form generator and other shared facilities.

The object level is the most interesting part of the MPCAL structure. MPCAL does everything by dealing with primitive and aggregate objects. Appointments, reminders and roles are primitive objects. Days, weeks, months, users and calendars are all aggregate objects. Aggregate objects contain primitive objects. For example, a day includes all reminders, and appointments that are part of that day.

Each object has its own second level module. The command loop at the top level reads a command and invokes the proper object module. That module in return does some processing and may call upon shared services from lower level modules. This structure makes it very easy to manage the code that applies to one object. The

object module is a set of action procedures that can manipulate the object. The "help" associated with each command is located with the object. This make it simple to change the help for a command when the procedure is changed in some way. This structure also allows the role facility to interface and control the calendar at the object level, while still remaining separate from the procedures it controls.

3.2.2 MPCAL's Role Facility

MPCAL improves computer based meeting scheduling by sharing calendar information and allowing delegation of authority. The role facility controls every function, including sharing and delegation of authority, in a simple, general and very precise manner. A role database supports this control activity. The calendar's support for conversations and the implementation of the role database are described in this section.

Every MPCAL user acts within a well defined set of capabilities. These calendar capabilities are assigned to individuals and referenced to specific calendars. They may be either explicitly or implicitly assigned to a person. Depending upon the calendar, an individual's capabilities may vary across a wide range of access to information and commands. For instance, a principal actor in one calendar may have no rights in another calendar.

MPCAL is capable of recognizing what people can see and do in various calendars. It recognizes the principal actors and the people authorized to act for

them. It understands when people can share information, and under what conditions a proposal may be directly attached to a calendar. To provide this support MPCAL requires extensive facilities for recognizing individual capabilities.

MPCAL also records significant calendar acts as they occur. For example, meeting proposals and confirmations are recorded. Principal users receive special notification of important actions. Each meeting has a history that includes information about its creation, and distribution.

The definition of a significant act may vary according to the person and calendar involved. For instance, when a calendar object is created it is always significant. On the other hand, when the details of an appointment are read the importance of the action varies with the situation and the person doing the reading. For example, if a secretary creates an appointment the date of creation is recorded as part of the appointment's history. When the manager sees the new commitment the notification is recorded in the history. If another person sees the appointment it may not be important, so the act is not recorded. The calendar needs a means for distinguishing what is significant and what is not.

Role Rules

MPCAL's ability to recognize individual abilities, share information, and record significant events is based upon specific rules that apply to every object and

operation in the calendar. These rules are combined into complete sets, one rule per object class and operation, and these sets are called roles. A *role* in MPCAL is a complete set of control rules. A person interacting with a particular calendar always acts within a role. When a command is invoked the role is checked to see if the command is allowed and whether or not special notification, highlighting or recording functions are needed. Roles are either specifically assigned to a user or a default calendar role may be given unknown users.

Individual rules are the foundation for all delegation of authority, information sharing, principal notification, change highlighting and significant event recording in MPCAL. They have several important characteristics. Each rule is a precise control mechanism that specifies the conditions for command access to MPCAL objects plus an arbitrary number of trigger [16] functions used for notification, highlighting and recording purposes. The rule structure is general enough to support every MPCAL function. Each rule is easy to understand and completely self contained.

```
(1) OPERATOR
(2) OBJECT CLASS
(3) ACCESS PREDICATES
(4) INTEGRITY CONSTRAINTS
(5) TRIGGER FUNCTIONS
```

Figure 3-1: Role Rule Structure

The structure of the role rule is precisely defined in MPCAL (Figure 3-1). There is one rule for every command in MPCAL. The rule includes an operation, an object class, a set of access predicates, integrity constraints and trigger functions.

Each rule is written for a specific object class. A *class* is a group of objects which are semantically similar. MPCAL object classes include appointments, reminders, roles, days, weeks, months, and calendars. Each individual object has content and a set of attributes. The content includes a unique identifier plus several named fields. For instance, an appointment includes an identifier, date, start time, end time, keywords, participant list and comments. The attributes of an object describe it in some way. Appointments have history, highlighting and notification attributes.

```
(1) OPERATOR: Change

(2) OBJECT CLASS : Appointment

(3) ACCESS PREDICATES:
    a) If the current user made this appointment
       allow change.
    b) If the current user is the PRINCIPAL
       allow change.

(4) INTEGRITY CONSTRAINTS:
    a) If changed appointment does not
       conflict with current commitments
       allow change.

(5) TRIGGER FUNCTIONS:
    a) Add history note to appointment.
    b) Highlight change to authorized actors.
```

Figure 3-2: Role Rule for Change Appointment Command

A rule exists to control every command in MPCAL. Figure 3-2 is an example. A command is a request for a specific operation to be applied to an object. Each rule contains a set of access predicates, and integrity constraints. The predicates specify when an operation can be attempted. The constraints are used to check the calendars semantic integrity before the operation is completed.

An operation's access to an object is controlled with content dependent predicates. MPCAL is a closed system [17], which means that it is necessary for access to be explicitly granted before an object can be operated upon. For instance, in a rule for appointment changes (Figure 3-2), the user must either be the creator of the meeting or a participant before any change is attempted. If one of these conditions is not met it is impossible to change the specific appointment.

A rule's access control can be very precise. It is possible to write a predicate based on any field of the object. General predicates can be written that always grant or always restrict access to an object class. Different types of access are also possible. For example, when a display operation is applied to an appointment either access to the entire content of the appointment is granted or only a limited outline is released.

Before an operation is completed semantic integrity constraints are checked. Integrity constraints are designed to maintain the correctness of the MPCAL database. Whenever an operation creates, changes or deletes an object all assertions (semantic constraints) on the database must hold true after the operation [17]. For

example, some calendars do not allow new appointments that conflict with other confirmed appointments. This constraint cannot be checked until the new object is completely defined. If there is a conflict the operation is rejected before it changes the database.

Role rules also contain trigger functions. These functions control notification, highlighting and recording in the calendar. They produce very important side effects. Immediately prior to completing the operation each listed function activates. If the trigger fails the operation also fails.

Precise control of recording, notification and highlighting in MPCAL is extremely important. Often a rule for a given command in different roles may be exactly the same except for the use of these functions. For instance, when a principal confirms a meeting proposal no notification is required, but if a different person confirms the appointment it is essential the principal be notified. Instead of burying the control for notification in each operation, control resides in individual role rules.

This completes the description of individual rules and their structure. This rule structure provides a flexibility that would not be available if all the access, integrity and trigger functions were built into each operation. Consolidating this control information in one place permits easy review and change. The rules provide precise control over calendar functionality.

MPCAL Roles

A role is a set of control rules. The set includes one rule for each command in the application. Even though several role definitions may exist in a calendar only one is active. This role defines the conditions each command must meet before being attempted. It also enforces semantic integrity, event recording, highlighting, and notification policy.

The behavioral science definition of a role as a set of behavior expectations based on individual identity and work context [7] closely parallels MPCAL's role concept. Each control rule defines an individual behavior and a command context. The role is a collection of these rules. Furthermore, the active role is always based on the identity of the current user. Indeed, MPCAL roles are roles in the more classical sense.

MPCAL uses roles for several reasons. First, a role is an effective abstraction for a complete rule set. A person does not have to deal with individual rules to understand a role. People have an intuitive feel for the meaning of a role and they are not interested in specific rules. Second, capitalizing on a user's role expectations makes the system easier to learn and use. Finally, roles can be tailored either by an organization or an individual to meet specific information sharing and delegation of authority requirements.

There are six predefined roles in the MPCAL version used by the M.I.T. Office Automation Group (See Figure A-15). There are owner, secretary, supervisor, group, public, and outcast roles in this version. These roles meet the needs of most new and casual users of the calendar. They allow new users to immediately start using the calendar without having to worry about a role definition. Casual users seldom require other roles. These standard roles also act as a common point of reference for group interaction.

MPCAL's role based control structure permits calendar tailoring. Standard roles are designed to match organizational expectations for behavior. They should reflect how most people coordinate meetings in the organization. These roles need to fit the specific organization. Therefore, it is unlikely the standard roles in the M.I.T. version of MPCAL would meet another organizations requirements.

It is very important that individual MPCAL users be able to create and modify roles. Standard role creation is significant for the organization so it is reasonable to expect a system developer to be involved. Roles defined by individuals are used by a limited audience and the organization should not be involved. There is too much overhead if people cannot meet their own requirements in a timely way.

Creating a new MPCAL role is straightforward. All it requires is a limited understanding of the current role capabilities. A new role must be uniquely named, given a short description and related to an existing role. A new role is created by

making an exact copy of the related role, changing the name and inserting the new description. Once a new role exists it can be changed one rule at a time.

The method for changing a MPCAL role takes time. It tries to ensure the user understands the options available, and it requires the change to be confirmed before it is completed. If the role is almost correct to begin with, only a few control rules may require modification. The MPCAL facility for changing roles is designed for "tuning up" a role, not changing every control rule at once. Changing a role is a gradual process. Each individual control rule in a role can be changed. A rule's access conditions, integrity constraints, trigger function list can all be modified.

Changing a control rule in a role definition takes several steps. First, a specific command is identified. The command includes the operation and object class of the rule. Next, the rule is displayed. The user can edit the display by choosing access predicates, integrity constraints and trigger functions that may apply in the rule. Once the rule is redefined a complete description of the rule is generated and must be confirmed before the new rule is installed in the role description.

Individual capabilities in MPCAL are controlled by the role the person is using. Users always interact with a calendar according to their active role. Each person is explicitly or implicitly assigned a role set in every calendar. The role set has at least a default role and may include several secondary roles.

Default roles control information sharing and determine the initial role a person uses to visit a calendar. During the process of creating a meeting proposal information may be extracted from several calendars. As each calendar is checked, the role of the person creating the proposal is checked to see if any information can be released. Whenever a person starts or visits a calendar a default role is assigned. If no explicit default role exists for a user the calendar provides a standard default (See Figure A-16).

Secondary roles may be assigned to each calendar user. These roles can be assumed at any time. Secondary roles allow users to switch their working context. For example, a person acting in a secretary role might assume a public role. The public role would limit the person's abilities. Secondary roles are also useful when a role is being assigned, created or changed. The role can be played before the action is taken.

The description of MPCAL roles is now complete. The roles provide a simple, easy to use, general and precise mechanism for controlling every function in MPCAL. In particular they make information sharing and delegation of authority possible. Without these two capabilities the functionality of MPCAL would be severely limited.

3.3 Evaluation

The MPCAL project has been a useful research vehicle. It explored computerized coordination support. Many problem seen in earlier calendar versions have been resolved. New issues came up and most were dealt with effectively. Some problems still await resolution. This section summarizes the lessons learned and highlights MPCAL's shortcomings.

3.3.1 Lessons Learned

Three significant lessons have been learned from MPCAL. They are:

1. Role based control is effective, flexible and precise.
2. Ad hoc and structured techniques for coordination are both necessary.
3. The system's basic functionality cannot be separated from features that support user interaction.

The role mechanism in MPCAL has worked very well. It controls access, enforces integrity constraints, and activates trigger functions in a consistent manner. The rule structure is general enough to support any command. Separating the control from the remainder of the application has made it possible to centralize role information and allows the effective modification of the role base.

Previous calendars supported an ad hoc approach to meeting coordination. This was fine, but MPCAL's "structured meeting conversations" has provided a complete method for creating, distributing, and responding to meeting proposals. This facility

has proven very useful.

The design of MPCAL's interface features cannot be separated from its functional design. Access to support information is always available and the user knows what the system is doing at all times. MPCAL's interface is integrated into the functionality of the system.

3.3.2 Areas for Improvement

Four areas require further attention. These areas are:

1. Communication reliability needs improvement.
2. Applications must share resources that optimize window management, form management and other common services.
3. Support for a wider range of predicates should be integrated into the system.
4. The calendar application should be tied into other applications such as a mail server or common text editor.

Moving messages in MPCAL is unwieldy. It is important to extract information from other calendars in real time, but the distribution of a proposal or response does not require this type of service. MPCAL gives up when it cannot immediately deliver a message. A mail system could be utilized to provide more robust service.

MPCAL is a large program. It is almost too large to be linked on a DEC-20 with CLU. MPCAL includes too many service functions, such as the window manager,

command parser and distribution system. Either a machine with a larger address space must be used or common services should run as separate processes with their own address space.

MPCAL implements a limited range of access predicates and integrity constraints. A more general method for defining these predicates is required. Organizations and users should have the ability to define new procedures that can be used by the application. The CLU programming language is a major stumbling block since it does not support run time linking.

MPCAL has not been tied into other applications. A common text editor could be utilized for example. Other means for notification, such as electronic mail, could be added fairly easily. The application could be improved by integrating it into a work station environment.

Chapter Four

Coordination Support

One of the primary goals of the MPCAL project is to produce a set of guidelines for the design and implementation of coordination enhancing software. Chapter Three described MPCAL's design. This chapter presents the justification for the design. First, a coordination model is presented. Next information sharing and delegation of authority are reviewed. Third, role theory is described as a foundation for recognizing individual capabilities on a computer system. Finally, an overview of application design completes this chapter.

4.1 The Structure and Design of MPCAL Conversations

"Structured meeting conversations" in MPCAL are similar to Flores' coordination model "conversations for action" [18]. MPCAL was designed without a knowledge of Flores' work. As people initially started to use MPCAL several problems with its conversation facility became obvious. The conversations were not complete. For instance, they did not provide return messages when a proposal was rejected or confirmed. It turns out "conversations for action" provide one model of structuring complete conversations. Several of MPCAL original shortcomings might have been avoided if some model of a conversation had been recognized. At least, it is interesting to consider one model, to see where it leads and to analyze its

shortcomings.

In the following pages, important terms are defined and the "conversations for action" model is summarized. After explaining the fundamental concepts, several example conversations illustrate the model and highlight the coordination process it supports. This model is not a panacea, but it does highlight critical coordination issues.

4.1.1 Conversations for Action

The basic functionality of MPCAL is similar to a model for coordination called "conversations for action". This model is based on a philosophy of language started by Austin [2]. These conversations follow a structured pattern of clearly defined linguistic utterances. These conversations are "the minimal unit for social interaction oriented toward the successful performance of action". It is important to understand this model, the ideas that support it, and the propositions it uses.

"Conversations for actions" take place when an organization acts. Every conversation starts in response to a perceived need. Without the need there is no conversation. This idea of need, this justification for organizational action, is called a *breakdown.*

When a breakdown occurs equipment becomes the focus of attention. Most people are unaware of the equipment they use to perform routine tasks. Its

"transparent" to them until something unusual happens. A secretary does not consider how a typewriter works as a letter is typed. An executive does not worry about a recorder as a report is dictated. If the typewriter stops typing or the recorder stops recording, then the equipment snaps into focus and a breakdown is perceived.

After a breakdown is seen by a person, several things take place before a "conversation for action" is initiated. First, the person becomes an actor, since he accepts the requirement to personally act or to cause some action. Second, some blame for the breakdown is assigned. Next, the actor evaluates the current situation, considers prior commitments and places a priority on the problem. A major problem may be broken down into smaller parts and each part may require separate action and a different priority. Once priorities are assigned, tools for dealing with the situation are necessary. If the tools are unavailable, inadequate, missing, or the actor does not have the ability to use them, the organization's "network of help" is called upon to assist with the breakdown.

Consider a short example of a breakdown [1]. Your supervisor has asked you to attend an important business meeting across town. You promise to go and the company provides you with a car to make the trip. As you drive along the car starts to pull to the right so you stop. At this point, the car is no longer "transparent" to you. Possible explanations for the trouble rush through your mind. The problem could be in the ball-joints, alignment, or tires. As you get out of the car you notice a flat tire and the other hypothesis are forgotten. Your first reaction is to "blame" the

tire. Then you blame yourself for not checking the car before leaving. You are forced to reconsider your current commitments and the possibilities for action. You could leave the car, but instead you try to fix the flat. You start looking for a jack and tire iron. You finally notice the car has no spare tire but a note in the trunk tells you to call the company's dispatcher for assistance. At this point you are ready to start a conversation. In order to fulfill your previous commitments, you phone the company's garage, then you talk with your boss and finally, you catch a taxi to the meeting. Notice in this example the pattern of assigning blame, reconsidering priorities, searching for tools and locating a "network of help".

Every conversation, resulting from a breakdown, follows a distinct pattern of utterances between a human speaker and a human listener. A conversation always starts with a request for action. A conversation always ends with either a rejection of the request, or a promise to fulfill the request within mutually agreed upon specifications.

"Conversations for action" may be simple or complex. A simple one might only include an initial request and a rejection or promise. A complex one may have any number of secondary conversations within it. Secondary conversations are generally used to clarify and negotiate the specifications of the original request. Every conversation must be complete, but individual conversations can follow many different patterns before reaching a conclusion.

The individual requests, promises, and rejections that comprise a conversation are highly structured and precisely defined. A *request* is sent from a human speaker and it is received by a human listener. It is similar to an MPCAL "proposal". It requires a future action, includes a time of completion, and describes the measurements used to judge satisfactory performance. A *promise* expresses a person's commitment to meet a specific request. An MPCAL commitment is essentially the same as a promise. A rejection is a promise not to meet a particular request.

For requests or promises to be valid they must be complete. If any element is missing the utterance is meaningless and therefore contributes nothing to the conversation. This is not to say every element is explicitly stated. When people have a common understanding of a situation it is only necessary to provide the information that is not "obvious".

Every conversation's goal is a promise. A rejection is useless because the original problem still has no prospective solution. On the other hand, a promise is valuable. It is a commitment to repair a breakdown. When a promise is made an individual believes the resources and time necessary to complete the action are available. Even though there is never an "iron clad" guarantee the action will be accomplished, a promise is sufficient reason to believe the action will be completed and other promises can be made on this assumption. When a promise is made "a person commits himself to the intelligibility, truth, sincerity, and appropriateness of what

he says. [18]"

This partial summary is sufficient. The concepts of *requests, promises, breakdowns* and *conversations* are all present in MPCAL's functional design. The MPCAL system addresses a particular kind of recurring breakdown: the need for a meeting.

4.1.2 Meeting Conversations

Now, two conversations are presented to illustrate the use of requests and promises in "conversations for action". In each example the situation is the same. Two managers, Susan and Arnold are talking face to face. Susan needs to discuss an Office Automation Project with Arnold. The need for a meeting is the primary breakdown or reason for having the conversation. Susan's goal is to extract a promise from Arnold. If he agrees to see her, Susan can make other commitments based on this promise. They are peers in an organization and each one is trying to cooperate with the other. Susan knows nothing about Arnold's schedule. Each person is fully aware of their own current work schedules.

Figure 4-1 shows a typical unstructured conversation. In statement (1) Susan opens the conversation with Arnold. Notice, the statement is not a request. It does not include a specific time, so it is not a complete request. Arnold only knows the subject and a general time of day for the meeting so there is no way he could respond with a promise at this point. Arnold sees the first statement as a breakdown

request is made and a promise is returned. The meeting will occur unless extraordinary circumstances arise. Susan can act based on Arnold's promise.

```
1) Susan: Let's meet in my office at 10am Friday to
          discuss the office automation project.

2) Arnold: How about 9am, instead?

3) Susan: Fine.
```

Figure 4-2: A Structured Conversation between Peers

The second conversation, in Figure 4-2, achieves the same result as the first. In this case, the conversation's structure is more efficient. (1) is a complete request. It includes all the essential information. (2) is a counter request that is exactly the same as the request in (1) except for the time. (3) is an acceptance by Susan of the counter request. It is a promise between Susan and Arnold to hold the meeting. Arnold's agreement to the meeting is implied in (2).

The scenario picked for these conversations eliminated many important issues. Peer level, face to face conversation are straightforward. Conversations over a computer system must consider more general issues. Information sharing and delegation of authority are necessary.

4.2 Information Sharing

Coordination can be enhanced if information sharing is allowed. To clarify this, consider the following short example. Once again, Susan and Arnold are trying to coordinate a meeting. This time Susan knows something about Arnold's schedule (i.e., Arnold plays golf on Friday).

```
1) Susan: Let's meet in my office at 9am Friday to discuss the
          office automation project.

2) Arnold: Fine.
```

Figure 4-3: A Conversation with Information Sharing

The conversation in Figure 4-3 is short and to the point. A clear request is made by Susan and Arnold immediately responds with a promise. No secondary conversations are necessary. In the previous example conversations, Susan preferred a meeting time of 10am, but it is never mentioned. In this case, Susan knew Arnold always plays golf on Friday at 10:30. She uses this information to make a request with a higher probability of acceptance. The request that Susan makes is feasible, based on the information she has about Arnold's schedule. Of course there is no guarantee Arnold will accept any request, but there is no need to discuss a request for a 10am appointment that will probably be rejected.

The idea of a feasible request is separate from whether a request is complete or

not. A *request* is feasible if it is based on the best information available to the sender about the receiver's commitments. In the earlier example conversations, the original requests were feasible because Susan did not know about Arnold's golf date. In the last example, if Susan had asked for a meeting at 10am, knowing Arnold had a conflicting commitment the request would have been complete, but infeasible. Making a feasible request never insures the request will be acceptable. Only the person who receives the request can decide to accept or reject it.

Sharing information reduces, but never eliminates, the possibility of secondary conversations. Creating a request without any information about the receiver's commitments is like taking "a shot in the dark". The lack of information increases the likelihood of secondary conversations. At the same time, it is unreasonable to expect perfect knowledge about other people's commitments. Secondary conversations will always be a necessary part of "conversations for action".

Why is it important that an initial request be as feasible as possible? In face to face conversations, there is usually no need to worry about such things. Secondary conversations can quickly change the original request into a mutually acceptable form. In an automated coordination system the situation is very different. The problems of time delay and lack of user presence are always making effective conversation more difficult. Secondary conversations are at least expensive, and sometimes impossible. They are hard to keep track of and time consuming. Information sharing is a fundamental requirement for automated conversation

support. Without some information sharing capability the coordination process becomes inefficient and unwieldy.

- Initial requests can be made more feasible if information about other people's commitments can be shared. The sharing cannot be uncontrolled or intrusive. Individual privacy should never be threatened. Individual privacy rights must be respected.

"Information privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information about them is communicated to others. [17]" There are two basic guidelines to consider when discussing privacy issues. First, a balance is required between the personal cost of divulging information and the benefits of sharing information. It is important that an individual understand why the organization or another person wants the information. Second, the information collection system must be fair. An individual needs to control the collection process, correct errors in information that has been collected and ultimately be able to prevent the system from making unwanted disclosures.

There may be some conflict between an organization's desire to share information freely and the individual's desire for privacy. Information sharing is an important part of an automated coordination process, but individuals do not want their own information used against them in a coordination process. A solution to the

conflict should neither eliminate the ability to share information during a coordination process, nor allow an organization to decree all information public knowledge. Either approach could ultimately destroy a coordination system that relies on shared information.

To summarize, while information sharing is desirable, unlimited sharing is not. Control is required whenever information is released to support a coordination process. Privacy guidelines must be respected. The organization's need to enhance coordination activities should be balanced with the individual's need for privacy. Mechanisms that allow controlled information sharing are necessary.

For information sharing to be controlled individual capabilities need recognition. It is unreasonable for people in an organization to have full access to each others information. In the previous conversation examples Susan and Arnold were always portrayed as cooperating peers in an organization. It did not seem too unreasonable that Susan knew about Arnold's golf matches. Arnold probably told her at one time or another and her knowledge did not threaten him. The situation might be very different if Susan was Arnold's supervisor in the organization.

Information sharing is often based upon personal identity and individual positions in the organization. For example, the amount and type of information sharing a colleague enjoys, will probably be different than the sharing allowed with a supervisor. This does not mean no information can be shared, it means different

levels of sharing should be possible. Susan, the supervisor, might be told by Arnold's secretary that he has a commitment at 10am Friday, while Susan, the colleague, might be told about the golf date. If Susan, in either position, asked about Arnold's weekend schedule the secretary might feign ignorance.

4.3 Delegation of Authority

Information sharing is not the only way to enhance a coordination process. The process can also be improved by recognizing that people often are allowed to act for other people in organizations. There can be a major difference between being authorized to act and being responsible for the action.

Two terms, responsibility and authority, need clarification. *Responsibility* is the obligation to give a satisfactory accounting for some action or state of affairs. In many organizations accountability is synonymous with this definition of responsibility. *Authority* is the legitimate ability to take some action. The two concepts are very different. Responsible individuals can be held accountable for an action even though they did not authorize it themselves. A person acting with proper authority often is not responsible for what happens.

An effective coordination system requires a facility that allows individuals to act for other individuals. There are several reasons for this. A single person may not be capable of managing every commitment. The President of the United States

certainly does not hear about all requests for meetings or promises made for him. The President has other things to worry about. A few people may have direct access to him, but even this access is very structured. The same idea applies in many organizations. Some people need authorization to act for other people, if only to keep the work load manageable.

Modern technology is contributing to a proliferation of junk information and requests for action. The courtesy copy often is no courtesy. Electronic mail can easily smother some people in the organization unless something is done to filter the traffic.

The solution may be to have the computer system act as a filter. Immediate rejection of requests from outside the organization can force the person making the request to use an alternate means of communication, such as mail. This technique can also be used within an organization to limit the capabilities of individuals who insist on sending junk. Often simple automatic message rejection is not a viable solution. The shear volume of valid requests may require some delegation of authority.

Additionally, the speed of the coordination process can be improved when several people share the authority to create and respond to requests for action. It is not surprising several people can do more than one person acting alone. Similarly having several people authorized to act reduces the problem of user presence.

Consider a situation where a manager is on an important business trip for a week. A request from the chief executive officer of the firm may arrive while the manager is absent. The manager's secretary should have the authority to act in this situation. The secretary knows the manager would accept the request, so the secretary should be allowed to confirm it.

Finally, a system that allows delegation of authority supports organizational specialization. Different people in organizations are often authorized to do different things. One person may be authorized to make meeting commitments for a firm's president while another person is authorized to make financial commitments. A single individual may be responsible for both activities, but never be directly involved.

When a person decides to delegate authority in a computer system there are special issues to be considered. First, the act of delegating authority should always be confirmed. It is important enough to always be double checked. Second, the system needs to notify responsible individuals when they are committed by somebody else. Finally, audit trails are necessary. They insure people who misuse their authority can be found.

Information sharing and delegation of authority are important functions that an automated coordination system should include. An easy to understand, abstraction mechanism for information sharing and delegation of authority is required. People

need to understand what they can do without being forced to learn individual rules of behavior. Roles meet this need.

4.4 Role Theory

The ruler rules, the minister ministers,
the father fathers and the son sons

-- Confusius

Recognizing individual capabilities on a system is a central issue in computerized coordination support. Most computer systems either ignore the issue or provide limited facilities for recognizing categories of users. For instance, most text editors treat each user exactly the same. On the other hand, operating systems often have built-in categories of users, such as "operators", "wheels" and common users. Coordination support requires more general, flexible and precise mechanisms for controlling individual behavior on a system. Role theory lends several important insights into this area.

There are five generally accepted propositions in role theory [7]. They are:

1. Role theorists assert that "some" behaviors are patterned and are characteristic of persons within contexts(i.e., form *roles*).

2. Roles are often associated with sets of persons who share a common identity(i.e. who constitute *social positions*).

3. People are often aware of roles, and to some extent roles are governed by the fact of their awareness(i.e.,by *expectations*).

4. Roles persist, in part, because of their consequences (*functions*)

and because they are often embedded within larger social systems.

5. Persons must be taught roles (i.e.,must be *socialized*) and may find either joy or sorrow in the performance thereof.

"A role is a set of behavior expectations based upon individual identity and work context [7]." The emphasis in this definition is on "set of behavior", "expectations", "individual identity" and "context". Each one of these is an important part of a role.

Roles are sets of patterned behavior. Each behavior must be observable. A behavior set is often used to characterize how people act in a given role. For example, a doctor is characterized as seeing patients, visiting hospitals, reading medical journals and sending out large bills. Doctors are expected do these things, so each behavior is part of the doctor role. Notice, the role does not include characteristics like hair color, shoe size or religious belief.

Roles only apply to people. For instance, fate and computers do not play roles. The number of people actually associated with a role can vary greatly. There are roles that extend across society and there are roles that apply to one person.

Role based behavior is often triggered by contextual cues. For example, a secretary in an office will usually answer a phone when it rings, while a manager will not answer unless the secretary is absent. Many roles are only defined within a certain context. The actual context may be based on many different factors. The time of day, physical surroundings, specific activity, or presence of another person

may all contribute to the definition of a context.

Roles are characteristic sets of behavior that people display in certain situations.They make individual behavior more predictable. If a person's current role is known, characteristic role behaviors are likely to occur. For instance, people at a political rally are expected to cheer for their candidate or give a standing ovation after an inspiring speech. Roles also restrict behavior in certain situations. For example, standing ovations are not usually given in church.

Two prerequisite conditions must be met before people can be expected to display characteristic role behavior. First, they must recognize the social situation. Second, they must "know" how to act in the situation. When these two conditions are met behavior is more predictable.

Role behavior is learned. People learn to act like doctors, clerks or managers. Socialization, the ability to act more effectively in society, teaches roles. Socialization helps people learn behaviors that accommodate other people. Formal education also teaches people how to act in certain situations. A lawyer, for instance, is taught in law school how to effectively present arguments to juries.

Learning a role takes time. Some role behaviors are very complex, and are difficult to remember initially. It may be necessary, when learning a role, to refer to an expert for assistance. With practice, the behavior patterns become internalized

and the role feels "natural" without any conscious thought.

Each person has many roles. An individual may act as a teacher at work, as a gardener on weekends, and as a pitcher on the local baseball team. Normally roles do not conflict because the recurring behavioral contexts are narrow enough to only require one role. People have little trouble switching roles. There is no conscious thought to the switching process. It is not difficult to be acting as a business manager at one moment, and a baseball fan the next.

Roles are constantly changing. They evolve and specialize over time. For example, a new secretary in an office arrives with some general expectations about the job. These general behavior patterns are a start, but as specific situations arise more specialized behaviors are learned. A supervisor may teach the secretary how to accomplish specific tasks either by providing detailed procedural instructions or by assigning the task and critiquing the results. Either way the secretary learns what to do when the task occurs in the future. As time passes the secretary knows from experience what the supervisor expects. Mutual expectations develop. Both the supervisor and the secretary make changes to their standard roles to reach an accommodation. In this case, both the secretary role and the manager role evolve.

It is misleading to think that roles determine all human behavior. They do not. Each person is an individual who may have many different roles. No matter how much is known about a person's roles it is impossible to predict how they will act in

many situations. In new situations roles generally do not apply or several roles may come into conflict. Roles are only guidelines for action. They may be very strong or very weak guidelines depending on the situation. Individuals ultimately decide on appropriate action, and a person's roles may or may not influence the final decision.

4.5 Why Use Roles in Coordination Systems?

There are several important reasons for building role support mechanisms into office systems. Roles make the systems easier to learn and use. They draw their power from the social expectations people have already learned. They allow the evolution individual capabilities. They can serve as an effective teaching mechanism. Finally, they can be used to control information sharing and delegation of authority.

Using roles in a computer supported coordination system capitalizes upon the internalized behavior expectations that people have for themselves and others. People use their role expectations everyday to help them define appropriate behavior in different situations. A secretary "knows" it is proper to open business mail for a manager and improper to open personal mail. A salesperson walking into an office "knows" it is proper to ask when a manager is free for a meeting, and it is improper to ask what the manager is doing at a specific time.

Roles also serve as an effective abstraction mechanism for complex sets of

behavior rules. It is not necessary for a person to be conscious of every specific behavior rule that is part of a role. For example, people who act as temporary secretaries do not need to ask for help every time they have to do something. They already have an idea about what a secretary can see and do in different situations. These expectations have been developed over long periods of time and they are valid until something changes them. An application that uses roles naturally uses these general expectations to simplify its use.

Furthermore, a coordination system that uses roles immediately takes advantage of peoples understanding that roles change and become more specific as social accommodations between various actors take place. This puts some burden on the system to provide easy means for modifying roles, but it supports the idea that individual roles do emerge. People understand there are differences between a generic manager role and the way Susan acts as a manager.

A system with built-in roles can help individuals learn about their capabilities in different situations. For instance, a new person in an organization may know very little about a secretary role when it comes to sharing a manager's calendar or making commitments for the manager. Using an automated calendar system will show the person what can and cannot be done. The organizational socialization process is being enhanced, since the system teaches simple role behaviors as tasks are performed.

Roles are particularly useful in recurring situations. In a coordination support application the context and the structure of the coordination process are limited. Most actions on the system occur very often. Only a few activities are supported at one time. For example, in MPCAL meetings are the primary concern of the system. Only two activities are supported. A means for carrying on conversations about meetings is provided and a structured display of meeting commitments and requests is presented. As long as the activities are well defined the role definitions are easily understood.

Roles can cross application boundaries. A work station might have a role management system that provides information sharing and delegation of authority capabilities for several programs. For example, a person's generic role does not necessarily change very much between a calendar system and an electronic mail application. Some role behaviors would be applied to several activities. At the same time, very application dependent behavior rules would be part of the system.

To summarize, roles are used in MPCAL because they make the system easier to learn and easier to use. They capitalize upon the knowledge people have about social capabilities and limitations. They provide a flexible mechanism for controlling information sharing and delegation of authority. Without roles it would be difficult to provide these capabilities in a simple integrated package. Indeed, roles are the central feature of a usable coordination system for the office.

4.6 Role Design

Practical experience with building roles into coordination systems has highlighted several important design considerations. Role names must be recognizable and distinguishable. Additionally, the design of standard roles must be tailored to an organization's policies and common expectations for behavior. Finally, methods for creating and modifying user defined roles are very important.

4.6.1 Predefined Organizational Roles

Standard roles are based upon observed organizational behavior. People act differently in different organizations, so it is important to study organizational behavior before building the standard roles. Once the context specific role expectations of an organization are defined, MPCAL can support these expectations.

Standard roles must be recognizable. A role is recognizable if its name evokes a set of behavioral expectations. These expectations, whether they are conscious or not, are extremely important. They make the system natural, easy to use and easy to learn. For instance, creating a "manager" role is useless, unless people have definite behavior expectations for "managers".

Standard roles should be distinct from each other. The behavioral expectations users have for each role must clearly distinguish the role. For instance, the behavioral differences between a secretary role and a public role must significant

and obvious. Practically speaking, the range of behavior is limited in MPCAL, therefore only a small number of individual roles can remain distinct. Three to six standard roles are sufficient in MPCAL.

4.6.2 User Defined Roles

In addition to the organization's ability to define standard roles, all MPCAL users can build their own role definitions. Standard roles, if properly researched and implemented meet the needs of most users. It is unlikely either new or casual system users will want to build new roles. On the other hand, people that use the system regularly often want to define and refine roles for their own use. MPCAL provides an integrated role definition facility for this purpose.

There are two fundamental reasons standard roles cannot sufficiently meet the needs of every user. First, they are generalizations of observed office behavior. They meet average expectations, but they cannot fit every need. A standard secretary role, for example, may include the authority to make commitments for the principal. Most people in an organization may expect this, but some may not. Second, while standard roles may work very well for a new user they need to evolve and become more specific over time. Individual capabilities change and the system must be flexible enough to keep roles current.

4.7 Designing Usable Office Applications

User interface issues have dominated the design and implementation of MPCAL. The interface is inseparable from the remainder of the system. In MPCAL, every design decision has been considered an interface decision. Every command, every screen display, and every error message, contributes to the development of the user's model of the system.

The primary concern of user interface design is not "ease of use", "ease of learning", or some ill defined notion of "user friendliness". The primary emphasis is on the factors that make the system "usable". A usable office application has four attributes:

1. It augments the effectiveness of managers and professionals in the office [41].
2. It meets and exceeds user expectations.
3. It is physically and conceptually compatible with a well defined user audience.
4. It reduces user anxiety, especially during initial encounters with the system.

4.7.1 Functionality First

Computer office applications should improve the quality of office work. In the past, "value-added" [41] applications were difficult to implement. They required communication and processing power that was not economically available.

Technology no longer stands in the way. Highly structured, efficiency oriented applications, such as word processing, are going to be supplemented with tools that serve the needs of managers, professionals and knowledge workers. These "value-added" applications are going to emphasize communication and coordination support activities.

The individual tools must be simple, consistent, and task specific. People interacting with computerized office systems are task oriented. They have a specific goal in mind, and they want to reach that goal with minimum effort. MPCAL is a key example. It manages conversations about meeting commitments. It is a calendar system and it is completely focused on that single application. By maintaining this focus, MPCAL has a simplicity and consistency that would be difficult to maintain in a more general application.

A personal workstation is an integrated set of individual tools. The applications must complement each other, even though they are conceptually separate. A person should be able to learn about a workstation's capability incrementally. Since each tool is functionally independent and internally consistent, it is possible to start with very simple tasks and gradually combine the tasks into complex personalized support systems. MPCAL, for instance, might be complemented with a time management application to produce a more generalized commitment manager.

Managers, professionals and knowledge workers need support for

communication and coordination activities. These people are action oriented, and they require a simple method for coordinating their work requirements. "Conversations for Action" are MPCAL's method. They are simple, structured and precise.

A usable coordination system also is a recognition of the communication limitations inherent in computer systems. Computer based conversations are not face to face conversations. Any model of automated coordination support must recognize reliability and user presence problems. The system must provide relevant, succinct, action oriented information during the coordination process. Information sharing and delegation of authority are practical necessities.

4.7.2 Exceed User Expectations

Once a "value-added" application is identified, the designer analyzes user expectations. The application must meet and exceed the user's needs. An uninformed designer's opinion does not mean very much. People always have an idea of what a computer should do. It is important an application designer understand how to meet specific needs and how to control unrealistic desires.

People resist changes to their working environment. To overcome this resistance a successful system immediately presents concrete advantages to a user. It does things naturally and it improves upon the process it replaces. It enhances the working environment.

Finding a natural model for an application is not easy. A "transparent" [12] model based upon organizational and individual expectations is required. Organizational expectations are typically reflected in policy and standard operating procedures. Individual expectations often center on the control structure of the system. In MPCAL, organizational needs are part of the default structure and individual desires for security are part of the control mechanism. MPCAL's physical model is a desk calendar, while its conceptual model is similar to "conversations for action".

One way to do more for the user is to encourage "problem mindedness" over "solution mindedness". There is always a tendency for the user to want the computer system to provide an "optimal" solution. In semi-structured situations, "optimal" solutions are not easy to find or may not exist. The system should try to turn problem situations into choice situations by generating alternatives for the user and letting him choose a solution. For example, information sharing in MPCAL only assists in the generation of meeting requests. The system guides, but never restricts a user's problem solving behavior.

4.7.3 Humanize the Interface

A usable application is physically and conceptually compatible with the people that use it. While the physical design of equipment is very important, it is usually outside the control of the typical application designer. On the other hand,

application designers do have direct control over a system's conceptual support facilities. Specifically, a designer determines the dialogue, help, and error prevention methods.

Dialogue Design

The design of the man-machine dialogue directly effects the usability of the system. When an application's interface is unwieldy, a common reason is a poorly designed dialogue. Because of this, dialogue design has been called the first step in the "user-centered design" process [31]. Specifying the method for information exchange is the first step in dialogue design.

Cheriton [9] has outlined the key issues in dialogue design. The command language should project a natural model to the user. This capitalizes on prior knowledge and gives a sense of recognition that contributes to the acceptance of the system. The dialogue should be centered on specific objects and the actions that affect them. Consistent relationships should exist between objects and the actions that operate on them. Finally, the language should be self documenting for the common user. This can be accomplished by using a vocabulary he can easily relate to the task.

Choosing the actual style of the dialogue is an important decision. Hebditch [31] talks of eight styles and Martin [37] outlines 21 different types. These range from

natural language and Query-By-Example to menus and forms. Flexibility is very important. As people initially learn about a system or as their needs evolve, a dialogue's style may change.

The concept of "closure" [37] adds another dimension to the dialogue. *Closure* occurs when a task is conceptually complete and the user is psychologically released to start another task. The intervals between closure points should be short. A single task should not exceed human short term memory capacity (5 to 9 steps or "chunks"). The system should take advantage of the closure points. They indicate times when the system can consolidate previous work without interfering with the human cognitive process. Immediately after task closure people usually hesitate, while they plan their next action. The system can use this time to finish processing. The system dialogue should clearly indicate closure points.

On-line Assistance

One of the major support features for any system is it's on-line assistance facility. On-line assistance replaces the hardcopy documentation found in manuals. The general attitude toward off-line documentation has become "The more (*off-line*) documentation to read, the fewer the readers [11]". For this reason on-line facilities have expanded significantly in recent years.

The on-line assistance must also take into account the relative needs of its users.

> The main criterion is to adjust the flow of interaction to the ability of the respondent. Nothing upsets a fluent speaker more than to be continually interrupted by comments, whereas a hesitant speaker may be grateful for the help. [34]

The system must recognize the character of its human dialogue partner. Each person will have individual needs and the computer should try to meet them on a case by case basis. In MPCAL different amounts of help are available depending on the style of dialogue being used.

Error Prevention and Correction

The prevention and correction of errors is a key topic in most human factors texts [38, 39, 40]. Errors can be divided into errors of omission, errors of commission, and sequencing errors. There are methods to correct or prevent each type. Errors can also be classified as idiosyncratic or situational [39]. Idiosyncratic errors reflect operator aptitude and motivation and they are influenced by skill and training. Situational errors are caused by poor procedures and incorrect training. Error frequency is related to task complexity and overload.

The fear of disastrous errors, where large a amount of work is lost or the system is brought "down", strongly influences the novice user. Nobody likes to lose work accidentally and most systems provide either a safety catch, by confirmation, or a reprieve, by undoing the operation. One of the first things a novice needs to know is he is always in control and apparent disasters can be dealt with easily.

Error prevention can be built into the interface. Using self-explanatory commands and natural syntax instead of codes and strange formats will help. Reasonableness checks can be used to localize errors as they occur. Defaults should be used when possible and they can be adapted to the users pattern of interaction.

4.7.4 Reducing User Anxiety

Whenever a person sits down to work on a computer system there is a certain amount of awe, fear, and uncertainty involved. Computers produce anxiety. The anxiety felt by a computer expert may be very focused, while new users tend to have a general sense of confusion and frustration. In any case, an application designer must recognize anxiety, and support individual efforts to cope with it.

People can be classified into several user types as they learn an application [6]. Novices are "uncertain" about the system. Novices hesitate to try new features and they view each dialogue step as a problem solving exercise [43]. By developing a mental model, novices gradually gain "insight" into an application. Infrequent users with insight are casual users. A frequent user, who continually uses an application in a task oriented manner, is an expert. Experts have "incorporated" the application and it is transparent to them.

Novice, casual and expert users share one common trait. They all are trying to complete a task using a computer tool. Each person is using the system in response to a "breakdown". The computer is seen as a tool, and as a network of help.

Novices require special attention. People are usually very anxious when they start to use a new application. They come with a problem to solve, but they are faced with a series of potential secondary breakdowns [50].

- "Reality shock" may occur when the computer model does not match the individual's view of the world.
- Confusion may set in and a novice may become disoriented and overwhelmed.
- Attempts to control the application environment often occur.

The system must recognize these potential breakdown areas and provide a "network of help" for each. If the breakdowns are not dealt with, the novice will reject the system before ever having any "insight" into its potential value. Secondary conversations associated with reality shock, confusion, and control attempts should be focused. Each secondary conversation is pure overhead, and too much overhead may make other methods more practical.

To summarize, the four attributes of a "usable" system are not independent of one another. For instance, the command language affects the conceptual model, and on-line assistance is largely designed to meet the novice's needs. The entire system fits together like a jigsaw puzzle. As long as any piece is missing the application is incomplete.

4.8 Summary

This chapter has sought to justify the design of MPCAL. It has touched upon a model for "conversations for action", the need for delegation of authority and information sharing, the use of role theory, and a general overview of designing "usable" systems. All of these things are important when integrating coordination support into automated information systems.

Chapter Five

Conclusions

5.1 Summary

A majority of office automation tools in the future will actively support organizational communication and coordination. Traditional applications, such as word processing and report generation, will be supplemented with structured communication aids. These new systems will develop the simple communication capabilities seen in electronic mail into more application oriented coordination support. This thesis explores the general character of these new tools and describes the design, implementation and use of a calendar coordination system.

The design of coordination support applications for the office is the central theme of this work. These applications will serve as the tools for organizational action in the future, and they require special consideration. People in offices coordinate to accomplish work, and computer systems can effectively support the coordination process. Computers can augment the definition and distribution of tasks and task related information within organizations. Coordination support applications can help organize task specifications, keep track of resource requirements, and generate up to date progress reports. These applications can help structure negotiations about task requirements and provide a dynamic history

mechanism that documents significant events in the coordination process.

This thesis goes beyond a discussion of design concepts. The concepts have actually been applied and tested during the implementation of the Multi-Person Calendar (MPCAL) coordination system. MPCAL supports structured coordination. It augments the process with information sharing and delegation of authority facilities. A unique role based control mechanism provides a simple, flexible and precise means for integrating organizational and individual expectations into the system.

5.2 Future Research

Research into coordination support should continue in several directions. Three of these are:

1. A method for identifying application specific behavioral expectations needs to be developed.

2. Coordination support applications, other then meeting scheduling, need to be implemented and critiqued.

3. The role mechanism should be extended to other applications.

Methods for integrating new applications into organizations are necessary. Effective coordination support applications must be tailored to the organizations they serve. Standard methods for identifying organizational expectations are needed.

Opportunities for research exist in other coordination applications. Meeting scheduling is a limited application. People coordinate in many ways. Joint document writing is another example. It is important to look at several applications and extract the underlying primitives of coordination support.

The structure of MPCAL's role facility could be used to provide flexible, precise control over other applications. It may be possible to build a more general role knowledge base that several applications share. This would reduce the overhead for each application and would provide and central repository of access rules, integrity constraints and trigger functions. The role base could be used to control both individual applications and their interactions.

Coordination support is a relatively new research area, so there are many opportunities for constructive research.

5.3 Conclusion

The primary goal of this thesis was to explore the design and implementation of automated coordination support applications. It has consolidated and extended previous work. Its design theory is based upon seemingly diverse disciplines of role theory, linguistic philosophy and the psychology of man-machine interfaces. These ideas have been brought together and applied to the implementation of a calendar coordination system. The implementation has highlighted many practical issues of

system design.

This work is neither the start nor the end of a long term research effort in the Office Automation Group at M.I.T. It has added more depth to the design theory behind coordination support. The MPCAL implementation is flexible enough to allow organizational testing and refinement of the role facility. Hopefully this work provides practical advise to system designers and a firm basis for further research.

Appendix A

MPCAL Screen Examples

```
 CIMRAL'S CALENDAR: VIEWED FROM JOYCE'S SECRETARY ROLE
   LAST COMMAND: SHOW DAY
TUESDAY        10 MAY 1983                  |       ** PROPOSALS **
                                            |
 9:00  EXAMPLE APPT                         |
 9:30  xx                                   |
10:00                                       |
10:30                                       |     No Pending Proposals
11:00                                       |
11:30                                       |
 NOON                                       |
12:30                                       |
 1:00                                       |-----------------------------
 1:30                                       |       ** REMINDERS **
 2:00                                       |
 2:30                                       |
 3:00                                       |
 3:30                                       |       No Reminders for Today
 4:00                                       |
 4:30                                       |
 5:00                                       |
                                            |

MPCAL>
```

Figure A-1: The Standard MPCAL Display

The MPCAL display as seen by an owner.
There are three separate display windows.

```
 CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
   LAST COMMAND: SHOW WEEK
                WEEK FROM 5 May 1983 TO 11 May 1983
-     THURSDAY |FRIDAY   |SATURDAY|SUNDAY   |MONDAY   |TUESDAY  |WEDNESDAY|
EARLY          |         |        |         |         |         |         |
 9:00    BUSY  | BUSY    |        |         | PROP    | BUSY    |         |
 9:30     xx   |  xx     |        |         |  xx     |  xx     |         |
10:00     xx   |  xx     |        |         |  xx     |         |         |
10:30     xx   |  xx     |        |         |  xx     |         |         |
11:00     xx   |  xx     |        |         |  xx     |         |         |
11:30     xx   |  xx     |        |         |  xx     |         |         |
 NOON     xx   |  xx     |        |         | PROP    |         |         |
12:30     xx   |  xx     |        |         |  xx     |         |         |
 1:00     xx   |  xx     |        |         |  xx     |         |         |
 1:30     xx   |  xx     |        |         |  xx     |         |         |
 2:00     xx   |  xx     |        |         |         |         |         |
 2:30     xx   |  xx     |        |         |         |         |         |
 3:00     xx   |  xx     |        |         |         |         |         |
 3:30     xx   |  xx     |        |         |         |         |         |
 4:00          |         |        |         |         |         |         |
 4:30          |         |        |         |         |         |         |
 5:00          |         |        |         |         |         |         |
 LATE BUSY     | BUSY    | BUSY   | BUSY    | BUSY    | BUSY    | BUSY    |
MPCAL>
```

Figure A-2: A Summary Display of a MPCAL Week

```
 CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
   LAST COMMAND: SHOW APPOINTMENT
TUESDAY   10 May 1983                  |       *** PROPOSAL ***
                                       |
 9:00 EXAMPLE APPT                     |   keywords:  EXAMPLE APPT
 9:30 xx                               |   date:       5-10-1983
10:00 xx                               |   from:      9:00am
10:30                                  |   until:     10:00am
11:00                                  |
11:30                                  |   participants:
 NOON                                  |      andrea
12:30                                  |      djc
 1:00                                  |
 1:30                                  |           ***  HISTORY  ***
 2:00                                  |  Appointment entered in this
 2:30                                  |  calendar on 28 April 1983 03:57
 3:00                                  |  by joyce. Flagged for
 3:30                                  |  confirmation. Proposal shown 28
 4:00                                  |  April 1983 04:05 to cimral.
 4:30                                  |
 5:00                                  |

MPCAL> show app 9
```

Figure A-3: Detailed Display of an Appointment

The history of the appointment is visible
when "SHOW APPOINTMENT" is used.

```
 CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
   LAST COMMAND: SHOW DAY
MONDAY    2 May 1983                    |      ** PROPOSALS **
                                        |
 9:00 MPCAL DEMO [New]                  |
 9:30 xx                                |  MPCAL DEMO [New]
10:00 xx                                |  5-2-1983      9:00   12:00
10:30 xx                                |
11:00 xx                                |  A Test [Canceled]
11:30 xx                                |  5-2-1983   12:00   14:00
 NOON *** [Can]                         |
12:30 **                                |
 1:00 **                                |---------------------------
 1:30 **                                |      ** REMINDERS **
 2:00                                   |
 2:30                                   |
 3:00                                   |
 3:30                                   |      No Reminders for Today
 4:00                                   |
 4:30                                   |
 5:00                                   |
                                        |

MPCAL>
```

Figure A-4: Highlighting Calendar Changes

New and Canceled appointments are highlighted until they are confirmed.

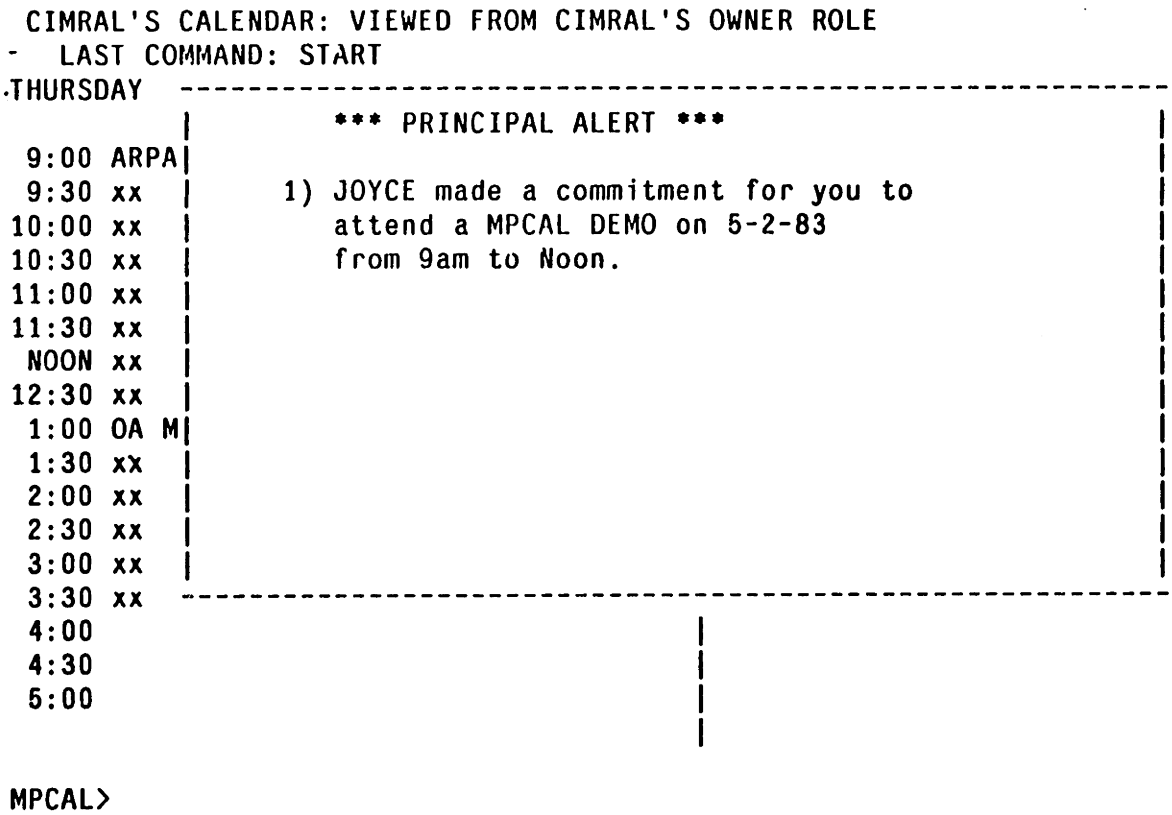

Figure A-5: Principal Notification Report

This report is presented to the principal calendar user when the MPCAL starts.

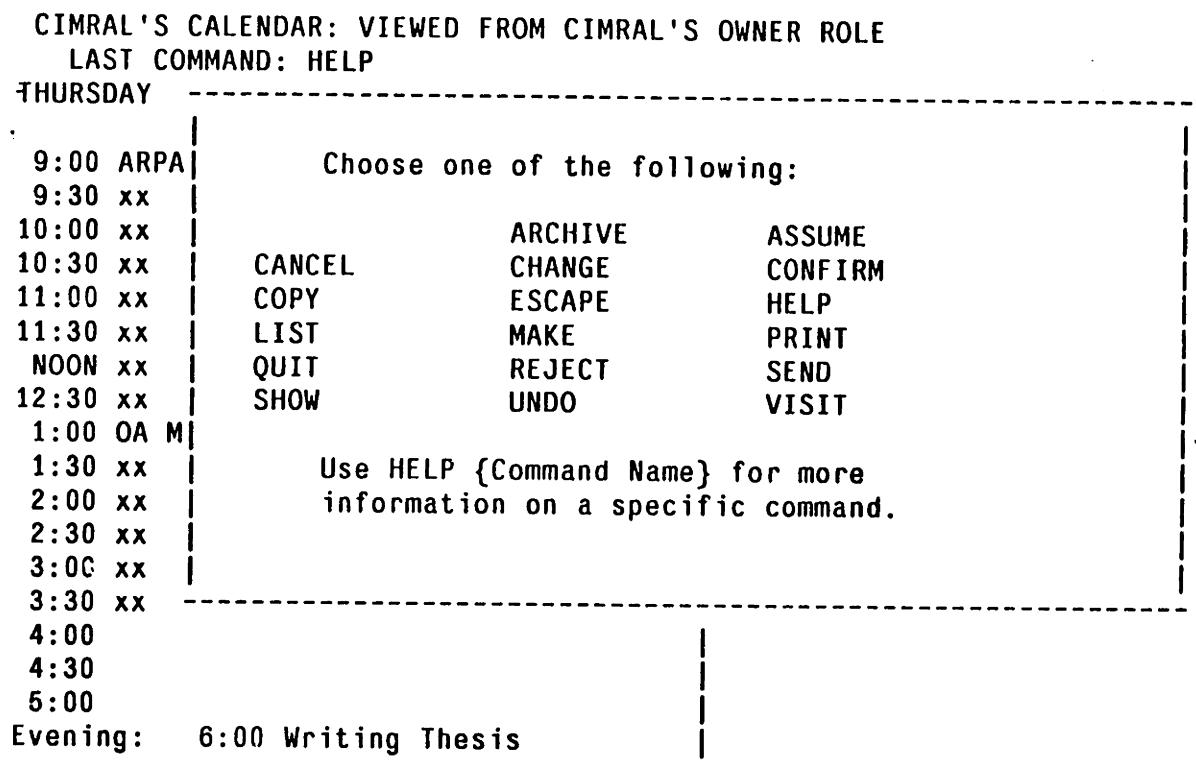

Figure A-6: MPCAL Commands

Each command may apply to several MPCAL objects.

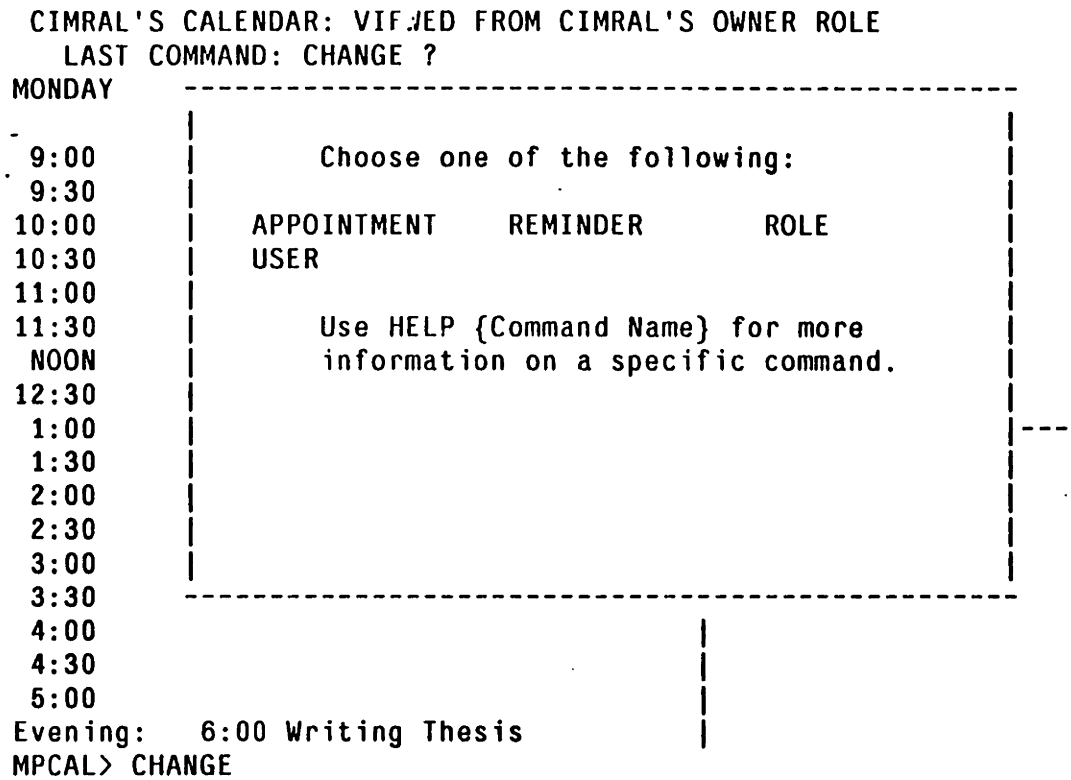

Figure A-7: The List of Objects "CHANGE" Can Manipulate

By entering a command followed by "?" a list
of objects the command applies to is available.

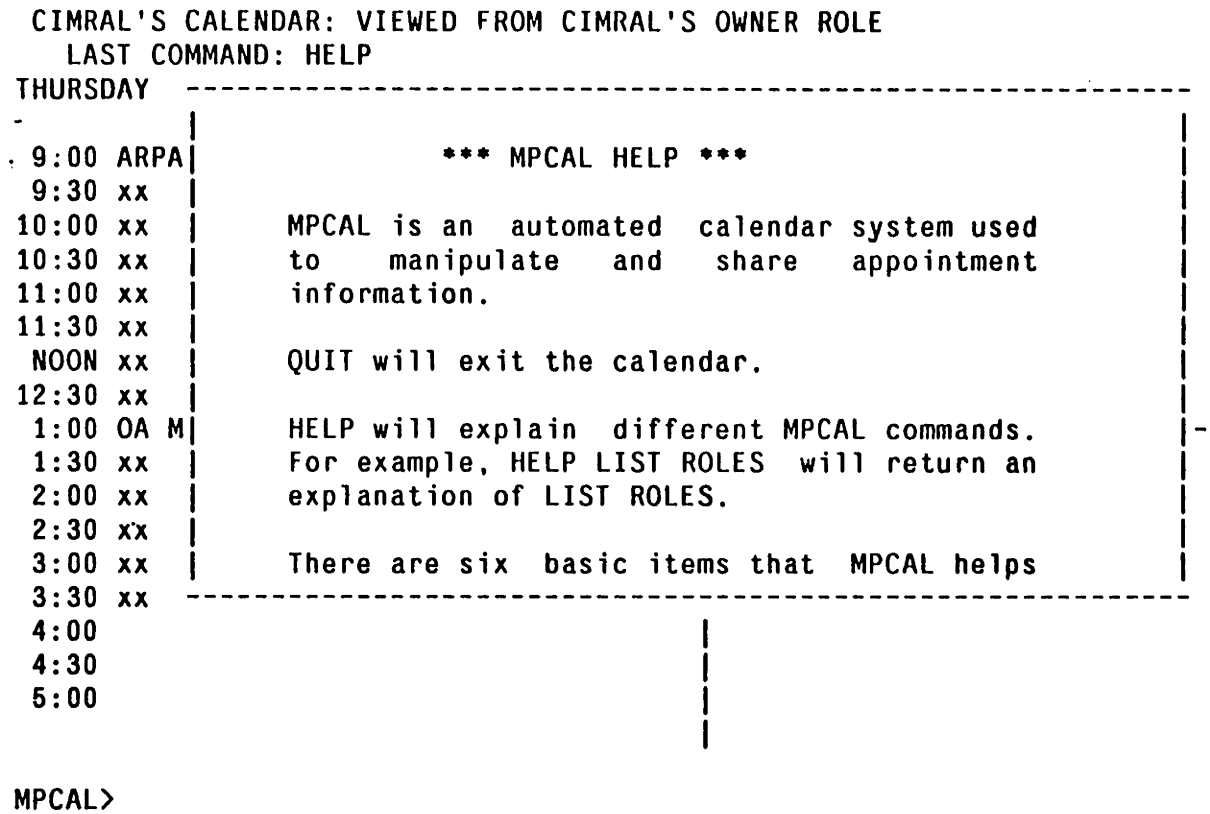

Figure A-8: MPCAL Help

This is the top level help message in MPCAL.
Each command has some help associated with it.
The help window scrolls to allow larger amounts
of information to be displayed.

```
 CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
   LAST COMMAND: MAKE APPOINTMENT
MONDAY    ------------------------------------------------------
          |      Start Time of Appointment (REQUIRED)           |
 9:00     |      FORMAT: 3:00pm or 3pm                          |
 9:30     |                                                     |
10:00     |      For help with the editor, type '↑H'            |
10:30     |      To insert a ?, type '↑Q?'                      |
11:00     |                                                     |
11:30     |                                                     |
 NOON     |                                                     |
12:30     |                                                     |
 1:00     |                                                     |--
 1:30     |                                                     |
 2:00     |                                                     |
 2:30     |                                                     |
 3:00     |                                                     |
 3:30     ------------------------------------------------------
DATE:5-9-1983   START TIME: (?)      END TIME:          KEYWORDS:
PARTICIPANTS:

COMMENTS:

MPCAL>
```

Figure A-9: Content Specific Help

Help for a field in a form is available
by typing "?" in the field.

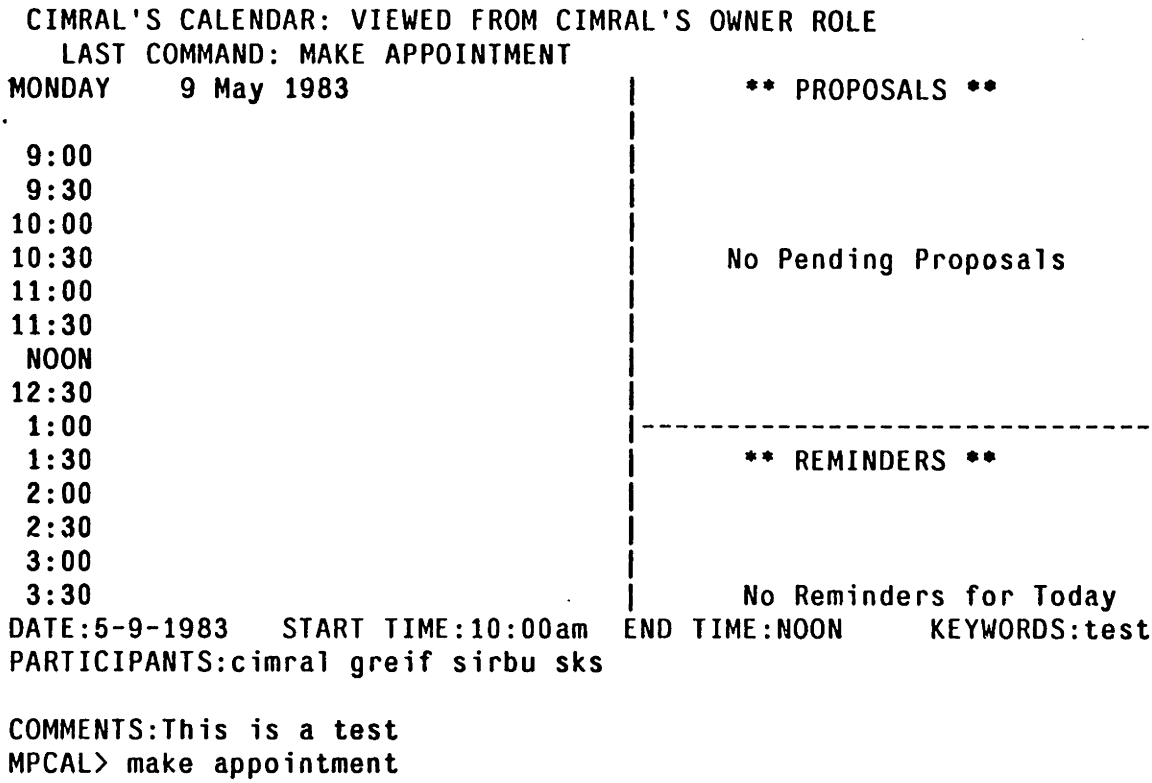

Figure A-10: A Command Form

```
 SIRBU'S CALENDAR: VIEWED FROM CIMRAL'S GROUP ROLE
   LAST COMMAND: VISIT
MONDAY    9 May 1983                    |      ** PROPOSALS **
                                        |
 9:00                                   |
 9:30                                   |
10:00                                   |
10:30                                   |     No Pending Proposals
11:00 *** BUSY ***                      |
11:30 xx                                |
 NOON                                   |
12:30                                   |
 1:00                                   |-----------------------------
 1:30                                   |      ** REMINDERS **
 2:00 *** BUSY ***                      |
 2:30 xx                                |
 3:00 xx                                |
 3:30                                   |      No Reminders for Today
 4:00                                   |
 4:30                                   |
 5:00                                   |
                                        |
                                        Now in SIRBU's calendar.
MPCAL>
```

Figure A-11: Visiting a Calendar

Note SIRBU's calendar will not release the titles of the appointments to CIMRAL.

```
CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
  LAST COMMAND: CHECK WEEK
              WEEK FROM 5 May 1983 TO 11 May 1983
  (1) : CIMRAL   (2) : SIRBU   (3) : GREIF  (4): SKS
    THURSDAY |FRIDAY   |SATURDAY|SUNDAY   |MONDAY   |TUESDAY |WEDNESDAY|
EARLY        |         |        |         |         |        |         |
 9:00  BUSY(1)| BUSY(2)|        |         | PROP(1)| BUSY(2)|         |
 9:30    xx   |   xx    |        |         |   xx    |   xx    |         |
10:00    xx   |   xx    |        |         |   xx    |         |         |
10:30    xx   |   xx    |        |         |   xx    |         |         |
11:00    xx   |   xx    |        |         |   xx    |         |         |
11:30    xx   |   xx    |        |         |   xx    |         |         |
 NOON    xx   |   xx    |        |         | PROP(3)|         |         |
12:30    xx   |   xx    |        |         |   xx    |         |         |
 1:00    xx   |   xx    |        |         |   xx    |         |         |
 1:30    xx   |   xx    |        |         |   xx    |         |         |
 2:00    xx   |   xx    |        |         |         |         |         |
 2:30    xx   |   xx    |        |         |         |         |         |
 3:00    xx   |   xx    |        |         |         |         |         |
 3:30    xx   |   xx    |        |         |         |         |         |
 4:00         |         |        |         |         |         |         |
 4:30         |         |        |         |         |         |         |
 5:00         |         |        |         |         |         |         |
 LATE         |         |        |         |         |         |         |
MPCAL>
```

Figure A-12: Checking a set of calendars

"CHECK WEEK" takes a list of calendars and creates a combined week display.

```
 CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
   LAST COMMAND: MAKE PROPOSAL
MONDAY    2 May 1983                    |  ***  PARTICIPANT CHECK  ***
                                        |
 9:00 MPCAL DEMO [New]                  |     This calendar checked.
 9:30 xx                                |     andrea - calendar checked
10:00 xx                                |     cal - refuses sharing
10:30 xx                                |     sks - calendar checked
11:00 xx                                |     djc - calendar checked
11:30 xx                                |
 NOON                                   |   TIMES WITHOUT CONFLICTS
12:30                                   |      8:30 - 9:00
 1:00                                   |      12:00 - 14:00
 1:30                                   |      15:30 - 17:00
 2:00                                   |
 2:30                                   |
 3:00                                   |
DATE:5-2-1983                   CHECK: x
PARTICIPANTS:andrea cal sks djc

START TIME:NOON     END TIME:2:00pm   KEYWORDS:A Test
COMMENTS:Please ignore this.

MPCAL>
```

Figure A-13: Creating a Request

When a request is being created a report
on participant calendars may be requested.
Time without conflicts are highlighted. Notice
that CAL refused to share information with CIMRAL

```
 CIMRAL'S CALENDAR: VIEWED FROM CIMRAL'S OWNER ROLE
   LAST COMMAND: SEND
MONDAY    2 May 1983                  |** SUMMARY OF PROPOSAL ACTIONS **
                                      |
 9:00 MPCAL DEMO [New]                |    Proposal sent to ANDREA.
 9:30 xx                              |
10:00 xx                              |    Cannot send to CAL.
10:30 xx                              |
11:00 xx                              |    Proposal sent to SKS.
11:30 xx                              |
 NOON A Test[New]                     |    Proposal sent to DJC.
12:30 xx                              |
 1:00 xx                              |
 1:30 xx                              |
 2:00                                 |
 2:30                                 |
 3:00                                 |
 3:30                                 |
 4:00                                 |
 4:30                                 |
 5:00                                 |
                                      |

MPCAL> Send 12 andrea cal sks djc
```

Figure A-14: Distributing a Request

When a request is distributed a report is immediately returned. It summarizes which calendars received the request.

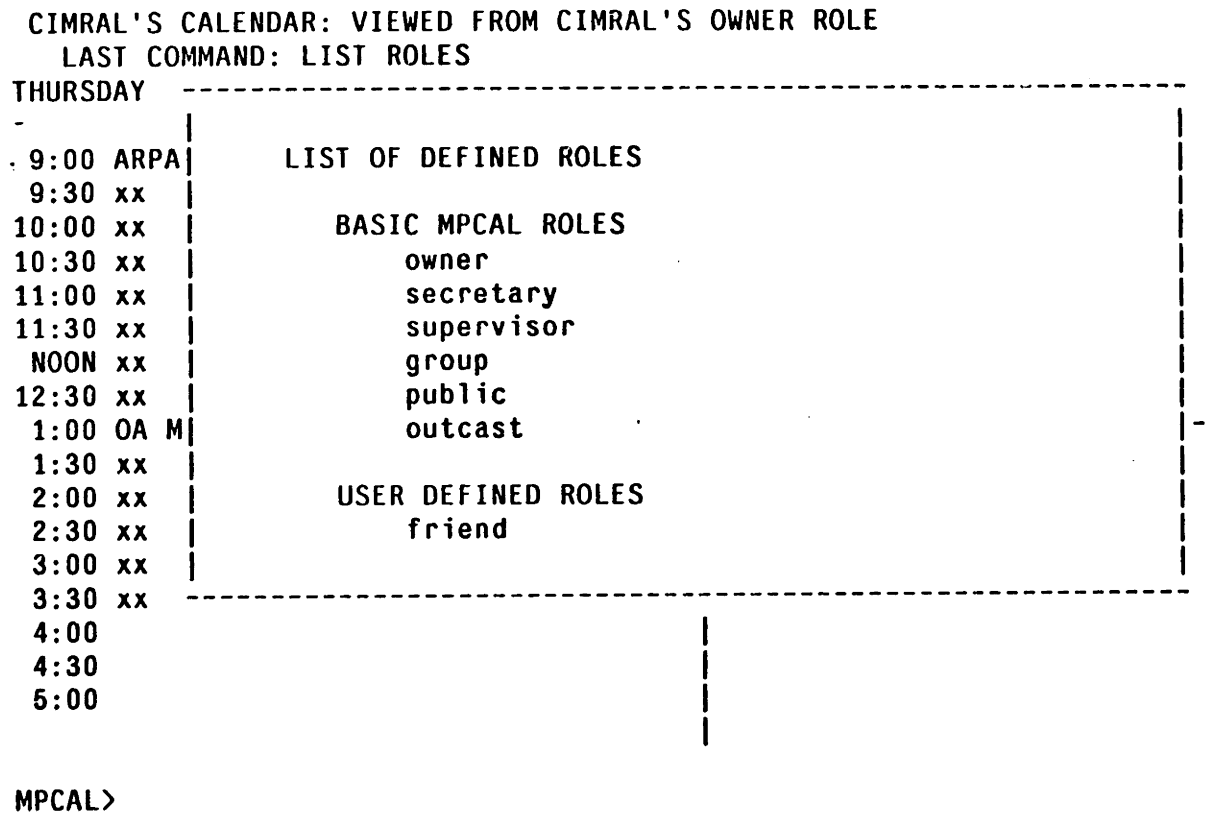

Figure A-15: List of Roles

There are six predefined roles in M.I.T's version of MPCAL. This calendar also has a user defined role of "Friend".

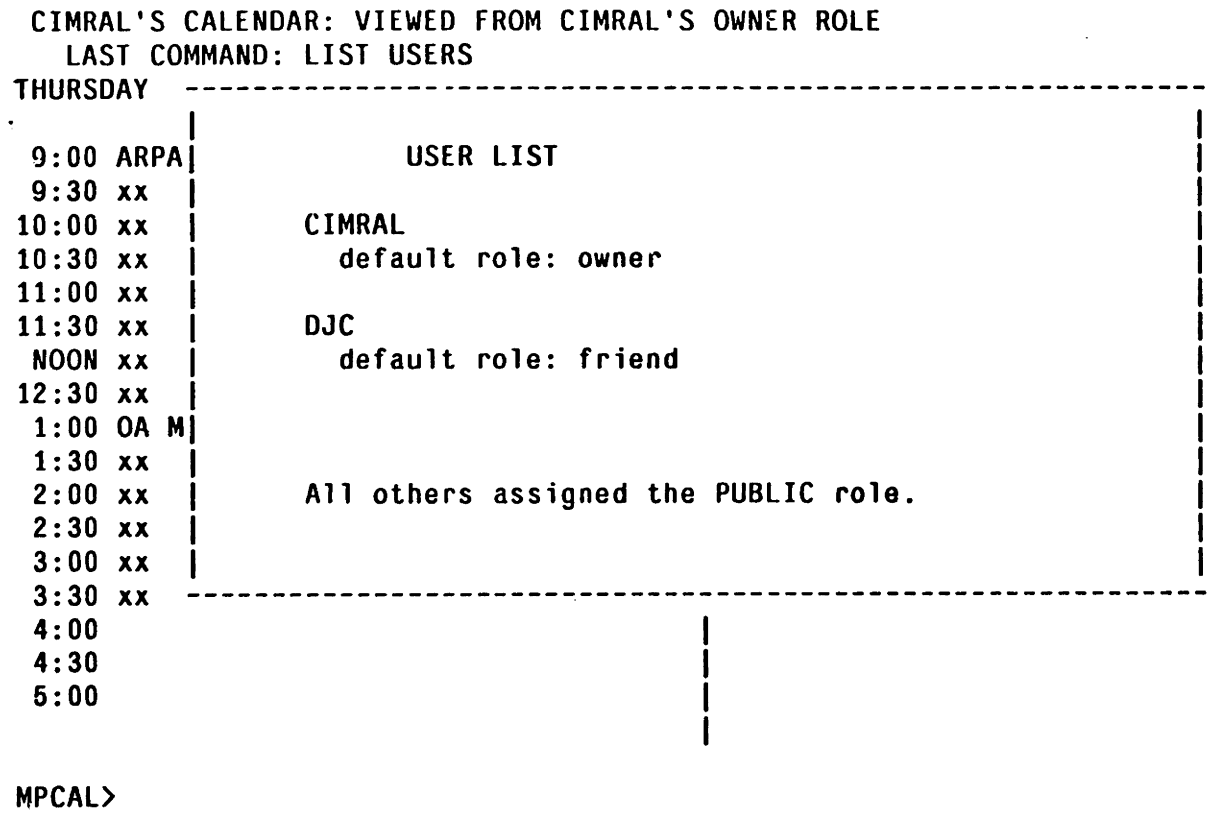

Figure A-16: List of Known Users

Each user is either explicitly assigned
a role or is given a default.

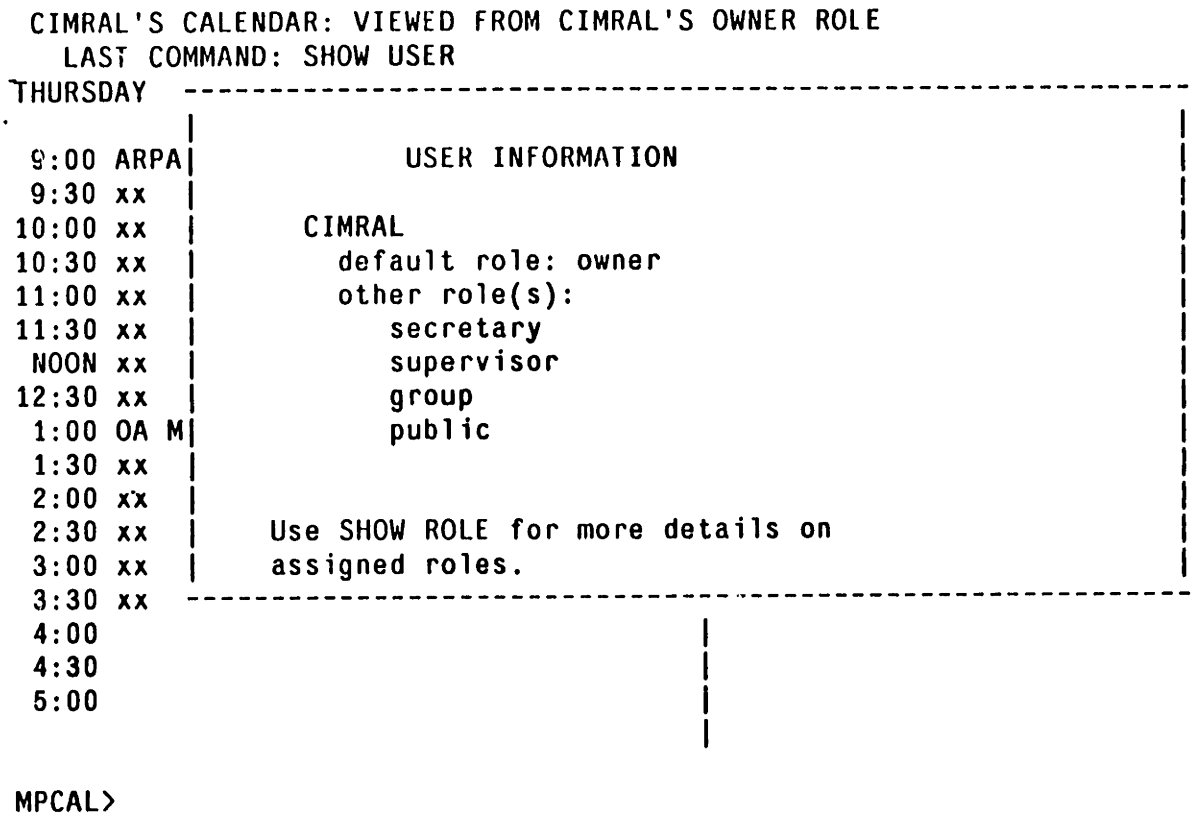

Figure A-17: An MPCAL User Description

Each system user is given a default role and
may optionally be assigned a set of secondary roles.

References

1. Aparo, Andrea. The Breakdown Process. Personal Communication.

2. Austin, John L. *Philosophical Papers.* Clarendan Press, Oxford, 1962.

3. Bair, James H. An Analysis of Organizational Productivity and The Use of Electronic Office Systems. BNR, Inc., July, 1980.

4. Barrett, Fred K. Communication through Shared Data. Massachusetts Institute of Technology Bachelor's Thesis. June 1981

5. Bennett, J. L. The Commercial Impact of Usabilty in Interactive Systems. In *Man/Computer Communication, Vol. 2,* Infotech State of the Art Report, Maidenhead, England, 1979.

6. Bennett, J. L. The User Interface in Interactive Systems. In *Annual Review of Infomation Science and Technology,* American Society for Information Science, Washington, 1972, pp. 159-196.

7. Biddle, Bruce J. *Role Theory: Expectations, Identities, and Behaviors.* Academic Press, New York, N.Y., 1979.

8. Card, S. K., Moran, T.P., and Newell, A. *The Psychology of Human-Computer Interaction.* Erlbaum, Hillsdale, N.J., 1981.

9. Cheriton, David R. Man-Machine Interface Design for Timesharing Systems. Proceedings of the ACM National Conference, ACM, 1976, pp. 362-366.

10. Conrath, David W. Measuring the Impact of Office Automation Technology Needs, Methods and Consequences. Proceedings of Office Automation Conference, Stanford University, Carmel, CA, 1980.

11. Cuff, R.N. On Casual Users . *Internation Journal of Man-Machine Studies 12* (1980), 163-187.

12. Dehning, W., Essig, H., and Maass, S. *The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs.* Springer-Verlag, Berlin, 1981.

13. Driscoll, James W. Office Automation: The Organizational Redesign of Office Work. Working Paper 106479, Alfred P. Sloan School of Management, MIT, May, 1979.

14. Driver, M.J., Streufert, S. Integrative Complexity: An Approach to Individuals and Groups as Information Processing Systems . *Administrative Science Quarterly* (Spring 1969).

15. Dzida, W., Herda, S. and Itzfeldt, W.D. User-Perceived Quality of Interactive Systems . *IEEE Transactions on Software Engineering SE-4* (1978), 270-276.

16. Eswaran, K.P. Specifications, Implementations, and Interactions of a Trigger Subsystem in an Integrated Database System. Report RJ 1820, IBM, San Jose, CA, 1976.

17. Fernandez, E.B, Summers, R.C., Wood, C. *Database Security and Ingegrity.* Addison-Wesley, Reading, Mass., 1981.

18. Flores, Carlos F. Management and Communication in the Office of the Future. Ph.D. Th., University of California, 1982.

19. Gaines, B. R., Hill, D. R. Man-computer communication - what next. 5th Man-computer communications Conf, International Journal of Man-Machine Studies, 1978, pp. 225-232.

20. Gaines, B. and Facey, P. Some Experience in Interactive System Development and Applications. Proceedings of the IEEE, Vol. 63, 1975, pp. 894-911.

21. Good, Michael. An Ease of Use Evaluation of an Integrated Editor and Formatter. Master Th., Massachusetts Institute of Technology, 1981.

22. Greif, Irene, and Hammer, Michael. Multi-person Information Work (Proposal to DARPA). Massachusetts Institute of Technology Office Automation Group Working Paper, WP-026. August 1980

23. Greif, Irene. Support Tools for Calendar Activities. Massachusetts Institute of Technology Office Automation Group Working Paper, WP-025. August 1980

24. Greif, Irene. *PCAL: A Personal Calendar.* Massachusetts Institute of Technology, 1981.

25. Greif, Irene. Cooperating Systems. Unpublished Proposal. August 1982

26. Greif, Irene. Software for the 'Roles' People Play. Massachusetts Institute of Technology Laboratory of Computer Science Technical Manual MIT/LCS/TM210. February 1983

27. Giuliano,Vincent E. The Hidden Productivity Factors of Office Information Systems. *Telephony Magazine* (July 1980).

28. Hammer, Michael, R. Ilson, et al. Etude: An Integrated Document Processing System. Proceedings of the 1981 Office Automation Conference, AFIPS, March, 1981.

29. Hammer, Michael M. and Marvin A. Sirbu. What is Office Automation? Proceedings of the National Computer Conference Office Automation Conference, AFIPS, March, 1980, pp. 37-49.

30. Hammer, Michael and Michael Zisman. Design and Implementation of Office Information Systems. Proc. NYU Symposium on Automated Office Systems, New York University Graduate School of Business Administration, May, 1979, pp. 13-24.

31. Hebditch, D. Design of Dialogues for Interactive Commercial Applications. In *Man/Computer Communication, Vol. 2*, Infotech State of the Art Report, Maidenhead, England, 1979.

32. Hsu, K. Sharing of an Office Calendar. Massachusetts Institute of Technology Bachelor's Thesis. May 1982

33. Kedzierski, Beverly I. Communication and Management Support in System Development Environments. *ACM Computing Surveys 13*, 1 (March 1981).

34. Kennedy, T.C.S. The Design of Interactive Procedures for Man-Machine Communication. *International Journal of Man-Machine Studies 6* (January 1974), 309-334.

35. Kennedy, T. C. S. Some Behavior Factors Affecting the Training of Naive Users of an Interactive Computer System. *International Journal of Man-Machine Studies* (1975), 817-834.

36. Kim, Y.J. Resource Sharing in an Automated Calendar System (PCAL). Massachusetts Institute of Technology Bachelor's Thesis. May 1982

37. Martin, James. *Design of Man-Machine Dialogues.* Prentice-Hall, Englewood Cliffs, N.J., 1973.

38. McCormick, E. J. *Human Factors in Engineering and Design.* McGraw-Hill, New York, 1976.

39. Meister, David. *Human Factors: Theory and Practice.* John Wiley and Sons, New York, 1971.

40. Meister, David. *Behavioral Foundations of System Design.* John Wiley and Sons, New York, 1976.

41. Meyer, N. Dean. The Office Automation Cookbook: Management Strategies for Getting Office Automation Moving. *Sloan Management Review* (Winter 1983), 51-60.

42. Mintzberg, Henry. *The Structuring of Organizations.* Prentice-Hall, Englewood Cliffs, N.J., 1979.

43. Moran, T. P. An Applied Physchology of the User. *ACM Computing Surveys 13*, 1 (March 1981).

44. Munford, E., Mercer, D., Mills, S., and Weir, M. The Human Problems of Computer Introduction. *Management Decision 10* (1972), 6-17.

45. Petit, Thomas A. *Fundamentals of Management Coordination: Supervisors, Middle Managers and Executives.* John Wiley and Sons, New York, 1975.

46. Prager, J. M. and Borkin, S.A. POLITE Project Progress Report. IBM Progress Report. April 1982

47. Roberts, Teresa L. Evaluation of Computer Text Editors. Ph.D. Th., Stanford, 1979.

48. Rohlfs, Sabine. User Interface Requirements. Convergence, Vol 2., Infotech State of the Art Report, Infotec, 1979, pp. 165-199.

49. Ben Schneiderman. *Software Psychology: Human Factors in Computer and Information Systems.* Winthrop Pub. Inc., Cambridge, Mass., 1980.

50. Sproull, L.S., Kiesler, S., Zubrow, D. Encountering an Alien Culture. CMU Department of Social Sciences Working Paper. Feb 83

51. Sutherland, Juliet B. An Office Analysis and Diagnosis Methodology. Massachusetts Institute of Technology Master's Thesis. Feb 83

52. Zdonik, Stanley. Object Management System Concepts: Supporting Integrated Office Workstation Applications. Ph.D. Th., Massachusetts Institute of Technology, 1983.

53. Zisman, M. D. Office Automation: Revolution or Evolution. *Sloan Management Review 19*, 3 (June 1978), 1-16.