AN OFFICE PROCEDURE FORMALISM USED FOR AN INTELLIGENT INTERFACE

COINS Technical Report 82-04

W. Bruce Croft
Lawrence S. Lefkowitz

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

A formalism for representing office procedures is presented. The emphasis in developing this formalism was on providing a representation which would enable the office information system to act as an intelligent assistant to the people carrying out the procedures. Two examples of the use of the formalism are described. The examples show that even common office procedures require sophisticated representations.

1. Introduction

A major issue in the design of an office information system (OIS) is the representation and use of routine office procedures. The tools commonly provided with the OIS software, such as the editor, electronic mail and electronic filing, implement and support the basic office tasks of information gathering, storage, communication, retrieval and analysis. These tasks are common to all organizations. An OIS, however, should also be used to support the more complex tasks that are the unique functions of a particular office. An example of this type of task would be the processing of an order form for a specific company. The steps involved in carrying out a task such as this can be called an office procedure. These procedures will be described using the procedure specification tool provided by the OIS. Using the information in the procedure descriptions, the OIS will be able to support and, to some extent, automate complex tasks in different environments.

The emphasis in this paper is on developing a formalism for representing office procedures that is flexible and powerful enough to enable the OIS to act as an intelligent assistant. This means that, amongst other things, the OIS should be able to recognize actions as being parts of procedures, carry out some steps automatically, suggest reasonable plans of action and warn the user of possible errors. This use of an office procedure formalism is to be contrasted with its use in the analysis and redesign of an office [ELLI80].

In the next section we shall describe in more detail how

office procedures can be used in the implementation of an intelligent interface. We shall then present, in outline, a formalism which was developed for this purpose and show two examples of its use in describing office procedures. These examples also point out some of the complexities which are encountered in a common office procedure such as arranging meetings. A more detailed explanation of the formalism and its use appears in [CROF82].

2. The OIS as an intelligent assistant

The descriptions of the procedures in the OIS will be used for two related purposes - interpretation and planning. Interpretation is the process of recognizing and understanding users' actions in the context of the office procedures. The ability to do interpretation allows the system to do the following:

1. Abstracting behavior. The procedures represented in the system will form hierarchies in which procedures located further up in a hierarchy represent more abstract tasks. The system will then be able to recognize a low-level action as being part of a procedure and describe the action in a more abstract fashion. For example, if a user filled out form 204A in a particular company, the system could recognize that this was part of processing an order.
2. Administrative functions. The system could keep track, over a number of terminal sessions, of a user's

activities. At any time, the users could ask for agendas of their activities which the system would present as partially completed procedures. The system could also monitor the progress of a procedure that a number of people may be working on and report its status to a manager.

Planning can be described as defining a sequence of actions that will get the system from an initial state to a goal state [FIKE72]. If an OIS has the ability to plan using the office procedure formalism, then it can:

1. Describe alternative plans of action to a user who is uncertain of what actions to perform for a particular procedure.
2. Carry out some parts of the procedures automatically. Most procedures require human interaction, but some actions in procedures such as filing copies of forms or calculating average sales figures could be carried out entirely by the system.
3. Follow the implications of a user's actions through a procedure and issue warning messages if the actions are inappropriate. A user may, for example, forget a step in a procedure or remove something that will be needed later.
4. Provide a higher-level interface. Because the representation contains a hierarchy of procedures at different levels of abstraction, the user may interact with the OIS at various levels. For example, the user may invoke abstract tasks, such as scheduling a meeting or

processing an order. These tasks consist of actions which may be carried out automatically by the system.

Interpretation and planning are related in that each requires the other in order to function efficiently. For example, interpretation requires a planning ability to help recognize the procedure to which a particular action belongs. A number of procedures may be initial candidates, but based on the user's objectives and the current context, the planner would indicate which candidates are more likely.

Incorporating the objectives of procedures into the representation formalism is essential for planning and for making the representation flexible enough to deal with real office procedures. The planner uses the objectives to determine the desirability of executing a particular procedure in order to satisfy a goal [SACE79]. Also, as Fikes pointed out [FIKE80], it would be unreasonable to expect that every possible way of carrying out a task could be specified in the system. The inclusion of objectives provides a means of recognizing that a task is complete even if the actual procedure used had not previously been specified.

3. The formalism and its use

The formalism selected must provide a convient means of describing procedures, their inter-relationships and their relationships to the information they manipulate. Work done on

an Event Description Language [BATE82] provides a basis for the desired formalism. A procedure is described in terms of primative operations carried out by tools in the basic OIS software (e.g. text editors, electronic mail, etc.) and in terms of other procedures. Thus, a hierarchy of procedures is developed with more complex ones based upon simpler ones. For example, a task such as preparing a letter may involve the use of a text editor, a spelling corrector and, finally, a text formatter. It is also possible to define abstract or generic procedures and then, by adding constraints, specific instances of a generic procedure. Using electronic mail may be considered a generic procedure, whereas using it to respond to an inquiry may be a more specific instance of the task.

The procedure description contains several distinct sections. The first section is the definition of the procedure in terms of other procedures. This portion, the IS clause, specifies the sequence of constituent procedures using the operators catenation (‘), alternation (!), shuffle (#), plus (+) and star (*). Catenation indicates that the occurence of the left operand procedure temporally procedes the right one. Alternation means that either the left or the right (but not both) operand procedure occurs. Shuffle allows the interleaving of the two operand procedures with their ordering being irrelevant. Plus means that the left (and only) operand procedure occures one or more times. Star is the closure of Plus and indicates zero or more occurrences of the procedure.

The second section of the description, the COND clause, contains constraints placed on attributes of the constituent

procedures specified in the IS clause. This allows a refinement of the procedural definition by specifying the conditions that must hold true if the procedure occurs.

The next section, the WITH clause, defines attributes of the procedure in terms of attributes of its constituent procedures. These attributes may then be referred to by other procedures by appending the attribute name to the procedure name. Hence, "procx.attriby" refers to the attribute "attriby" defined in the WITH clause of procedure "procx".

The PRECONDITION clause specifies a set of conditions, based on the attributes of entities in the OIS's database, that must be true in order for the procedure to begin. This is used by the recognition section of the system to determine if a particular procedure can be instantiated at a given time and by the planning section to ascertain what must be done in order to reach a state that would allow a desired procedure to occur.

The SATISFACTION section provides a means of recognizing the completion of a procedure. It contains a set of conditions that must hold if the procedure has finished. While similar to the COND clause, this section refers to attributes of entities in the OIS database and is meant to be independent of the method used to carry out the task. This enables the recognition of the completion of a procedure even it was executed by a means not specified in the IS clause of its definition. For instance, a procedure may define a typical means of acquiring a piece of information, but the information may be entered into the database because the user knows it a priori. The SATISFACTION clause would enable the system to recognize the "acquisition" task as

being completed even though the usual steps were not carried out. This section may also be used by the planner to determine what effects the execution of a procedure will have on the state of the entities in the database. If a SATISFACTION clause is empty, the completion of the procedure is dependent upon the completion of the sequence of constituent procedures in the IS clause.

4. Examples

Two examples of the use of the office, procedure formalism are presented in this section. The first example is the journal editing process described by Zisman [ZISM78]. The second involves the task of arranging a meeting. The journal editing process is relatively straight-forward with few alternative methods provided for accomplishing the objectives. The second example presents the greater challange because of the inherently greater flexibility of the task.

In order to simplify the writing of complex task descriptions, it is desirable to create a collection of relatively simple procedures commonly used in the particular application domain. A subset of such useful procedures for an OIS is presented in Appendix A. These routines describe, in the previously discussed formalism, some basic means of storing, retrieving, transmitting and receiving information. They form a basis for many of the higher-level tasks to be performed in an automated office environment.

The journal editing procedure consists of, in brief,

receiving a paper, acknowledging its receipt, asking (and reminding, if necessary) the editor to select referees, asking (and reminding) the referees to review the paper, replacing the referees if they cannot review it, collecting the reviews (reminding the referees, if necessary), asking (and reminding) the editor to make a decision when all the reviews are in, and informing the author and the executive editor of the decision. The description of the top level procedure to carry out this task appears as Figure 1. The lower level procedures required are contained in Appendix B and in the basic office procedures in Appendix A.

The task of arranging a meeting requires that the user (i.e. the person requesting the meeting) select the people s/he wishes to attend, the possible time-slots in which the meeting may occur and, if desired, the set of rooms in which it may be held. Then the desired attendees must be surveyed to find out when, in the *selected* time-slots, they are available. If there is no time-slot in which all attendees are free, then the user must reconsider her/his choice of attendees and/or time-slots. When one or more mutually acceptable time-slots are available, a suitable (and available) room must be selected. Once this is done, the attendees are informed of the meeting place and time and asked (with reminders, if necessary) to confirm that they can attend. If they cannot, the time-slot/room selection process is repeated until a suitable time and location are found or until all available choices have been exhausted. If the latter occurs, the user will again be asked to reconsider her/his choice of attendees and/or time-slots and the selection process repeats.

When a suitable choice is found, final notice is sent to the attendees and the room reservation is confirmed.

Figure 2 provides the top level procedure for arranging a meeting, with the supporting procedures in Appendices A and C. Note that while the task is considerably more complex than the journal editing procedure, the representation in the office procedure formalism, though somewhat more intricate than the first example, is still fairly straight-forward.

Description of a Journal Editing Process

```
PROC        Journal_editing

   description      The process of editing a journal paper as described
                    by Zisman.

   IS               Receive_paper ' [ { (Acknowledgement_to_author
                                          # Get_referees_from_editor)
                                     '  Referees_review_paper
                                     '  Get_decision_from_editor
                                     ' (Inform_author # Inform_exec_editor )
                                        }  !  Author_withdraws ]

   COND             Receive_paper.Author = Acknowledgement_to_author.author
                                         = Inform_author.Author
                                    OR   = Author_withdraws.Author

   PRECONDITIONS    -

   SATISFACTION     Author.decision NOT NIL
                           OR
                    Author.withdraws IS TRUE

   WITH             Author     = Receive_paper.Author
                    Paper      = Receive_paper.Content
                    Referees   = FOR ALL i: Referees_review_paper.Referee(i)
                    Reviews    = FOR ALL i: Referees_review_paper.Report(i)
                    Editor     = Get_referees_from_editor.Editor
                    Executive  = Inform_executive_editor.Exec
                    Decision   = Get_decision_from_editor.Decision
```

Figure 1

Description of a Meeting Arrangement Process

```
PROC        Arrange_meeting

  description     Organize a meeting at a time and place acceptable to
                  the person requesting the meeting and the attendees.

  IS              (Get_attendees # Get_rooms # Get_times)
                    ' Get_attendee_availability
                    ' [Modify-choices '
                          (Select_room ' Check_with_attendees)+ ]+
                    ' (Confirm_room # Confirm_attendees)

  COND            Get_attendees_availibility.Attendees
                        = Get_attendees.Attendees
                  Select_room.Selected-room = Confirm_room.Room
                        = Check_with_attendees.Room
                  Select_room.Selected-time = Confirm_room.Time
                        = Check_with_attendees.Time

  PRECONDITIONS   -

  SATISFACTION    Meeting.Arranged IS COMPLETE

  WITH            Room      = Confirm_room.Info.Room
                  Time      = Confirm_room.Info.Time
                  Attendees = Confirm_attendees.Attendees
```

Figure 2

5. Conclusion

The examples show that the formalism presented in this paper can conveniently represent complex office procedures. It also provides information for planning and interpretation. Incorporating procedures described in this way into an office information system will enable it to act as an intelligent assistant in the execution of office tasks.

Acknowledgement

This work is part of a larger research effort examining Procedure Oriented Interfaces for Supportive Environments (POISE). Also involved in the project are Victor Lesser and Karen Huff, who are applying this approach to the software development environment.

References

[BATE82] Bates, P.C.; Wileden, J.C. "EDL: A basis for distributed system debugging tools." Hawaii Conference; 1982.

[CROF82] Croft, W.B.; Huff, K.E.; Lefkowitz, L.S.; Lesser, V.R. "POISE technical report" (to appear).

[ELLI80] Ellis, C.A.; Nutt, G.J. "Office information systems and computer science." ACM Computing Surveys, 12: 27-60; 1980.

[FIKE72] Fikes, R.E.; Hart, P.E.; Nilsson, N.J. "Some new directions in robot problem solving." In Machine Intelligence, , 7: 405-430; Edinburgh University Press, 1972.

[FIKE80] Fikes, R.E.; Henderson, D.A. "On supporting the use of procedures in office work." Proceedings of the First AAAI Conference, Stanford University, 1980.

[SACE79] Sacerdoti, E.D. "Problem Solving Tactics." Sixth International Joint Conference on Artificial Intelligence, Tokyo, 1979.

[ZISM78] Zisman, M.D. "Use of production systems for modeling asynchronous, concurrent processes." In Pattern-Directed Inference Systems, Academic Press, 1978.

APPENDIX A

BASIC OFFICE PROCEDURES

PROC Record_information

description	Places information into a database.
IS	primative
WITH	Info = information placed into the database Topic = a brief description of the information

PROC Look_up_information

description	Retrieves information from a database.
IS	primative
WITH	Info = information retrieved from the database Topic = a brief description of the information

PROC Send_a_message

description	Transfers information from the user.
IS	primative
WITH	Recipient = the person to whom the message is sent Topic = a brief description of the message Content = the body of the message

PROC Receive_a_message

description	Transfers information to the user.
IS	primative
WITH	Sender = the person from whom the message is received Topic = a brief description of the message Content = the body of the message

PROC Tell_information

description	Conveys information that resides in the user's database to someone else.
IS	Look_up_information ' Send_a_message
COND	Look_up_information. Info = Send_a_message. Content
PRECONDITIONS	Info NOT NIL
SATISFACTION	Informed. Info NOT NIL
WITH	Info = Look_up_information. Info Informed = Send_a_message. Recipient Topic = Send_a_message. Topic

PROC Be_told_information

description	Receives information and stores it in the user's database.
IS	Receive_a_message ' Record_information
COND	Receive_a_message. Content = Record_information. Info
PRECONDITIONS	-
SATISFACTION	Info NOT NIL
WITH	Info = Record_information. Info Informer = Receive_a_message. Sender Topic = Receive_a_message. Topic

PROC Request_information

description	Asks for and receives information.
IS	Send_a_message ' Receive_a_message
COND	Send_a_message. Topic = Receive_a_message. Topic Send_a_message. Recipient = Receive_a_message. Sender
PRECONDITIONS	-
SATISFACTION	-
WITH	Requestee = Send_a_message. Recipient Topic = Send_a_message. Topic Query = Send_a_message. Content Reply = Receive_a_message. Content

PROC Remind

description Sends information regarding a previously mentioned topic.

IS Send_a_message

COND Send_a_message.Topic.Creation-time <
Send_a_message.Start-time

PRECONDITIONS Topic NOT NIL

SATISFACTION -

WITH Recipient = Send_a_message.Recipient
Topic = Send_a_message.Topic
Content = Send_a_message.Content

PROC Find_out_information

description Asks for, obtains and stores information.

IS Request_information ' Record_information

COND Request_information.Reply = Record_information.Info

PRECONDITIONS -

SATISFACTION Info NOT NIL

WITH Requestee = Request_information.Requestee
Topic = Request_information.Topic
Query = Request_information.Query
Reply = Request_information.Reply

PROC Supply_information

description Answers a request for information.

IS Receive_a_message ' Look_up_information ' Send_a_message

COND Receive_a_message.Topic = Look_up_information.Topic
= Send_a_message.Topic

Receive_a_message.Sender = Send_a_message.Recipient

PRECONDITIONS -

SATISFACTION Informed.Info NOT NIL

WITH Informed = Send_a_message.Recipient
Info = Send_a_message.Content
Request = Receive_a_message.Content
Topic = Receive_a_message.Topic

```
PROC    Request_with_reminder (reminder-time)

        description     Asks for information and sends reminders every
                        "reminder-time" until the information is received.

        IS              Send_a_message ' Remind* ' Receive_a_message

        COND            Remind(1).Time >= Send_a_message.Time + Reminder-time
                        Remind(i+1).Time >= Remind(i).Time + Reminder-time
                        Remind.Topic = Send_a_message.Topic
                        Remind.Recipient = Send_a_message.Recipient

        PRECONDITIONS   -

        SATISFACTION    Info IS RECEIVED

        WITH            Requestee = Send_a_message.Recipient
                        Topic     = Send_a_message.Topic
                        Query     = Send_a_message.Content
                        Reply     = Receive_a_message.Content

PROC    Confirm

        description     Confirms previously transferred information.

        IS              Tell_information

        COND            Tell_information.Topic = "Confirmation"

        PRECONDITIONS   Confirmed.Info NOT NIL

        SATISFACTION    Message IS SENT

        WITH            Info      = Tell_information.Info
                        Confirmed = Tell_information.Informed

PROC    Task_complete

        description     Gets told that the execution of task is completed.
                        This is useful for determining the termination of any
                        of a class of open-ended tasks (such as adding an
                        unspecified amount of information to a database).

        IS              Be_told_information

        COND            Be_told_information.Topic = "Completion"

        PRECONDITIONS   Task NOT NIL

        SATISFACTION    Task IS COMPLETE

        WITH            Task     = Be_told_information.Info
                        Informer = Be_told_information.Informer
```

PROC Obtain_information

description	Obtains (with or without asking for) and stores information.
IS	Find_out_information ¦ Be_told_information
COND	–
PRECONDITIONS	–
SATISFACTION	Info NOT NIL
WITH	Info = Find_out_information.Reply or Be_told_information.Info Topic = Find_out_information.Topic or Be_told_information.Topic

PROC Select_choices

description	Gets information by having someone select one or more of a set of options presented to him/her.
IS	Find_out_information
COND	Find_out_information.Query.Type = "Choose-from"
PRECONDITIONS	–
SATISFACTION	Choices NOT NIL
WITH	Requestee = Find_out_information.Requestee Topic = Find_out_information.Topic Query = Find_out_information.Query Reply = Find_out_information.Reply

PROC Update_information

description	Obtains information about an already existing entity.
IS	Obtain_information
COND	Obtain_information.Topic.Creation-time < Obtain_information.Start-time
PRECONDITIONS	Info NOT NIL
SATISFACTION	Info IS MODIFIED
WITH	Info = Obtain_information.Info Topic = Obtain_information.Topic

APPENDIX B

JOURNAL EDITING PROCEDURES

PROC	Receive_paper	
	description	Obtains and stores a potential journal paper.
	IS	Be_told_information
	COND	Be_told_information.Topic = "Journal Paper"
	PRECONDITIONS	-
	SATISFACTION	Author.Paper NOT NIL
	WITH	Content = Be_told_information.Info Author = Be_told_information.Informer

PROC	Acknowledgement_to_author	
	description	Sends a message to an author acknowledging the receipt of a paper.
	IS	Send_a_message
	COND	Send_a_message.Topic = "Acknowledgement"
	PRECONDITIONS	(Author.Paper NOT NIL) AND (Author.Acknowledged IS NIL)
	SATISFACTION	Author.Acknowledge NOT NIL
	WITH	Author = Send_a_message.Recipient

```
PROC    Get_referees_from_editor

        description     Asks the editor for a list of referees to review
                        a paper.  A reminder is sent every two weeks until
                        the editor responds.

        IS              Request_with_reminder (2 weeks)

        COND            Request_with_reminder.Topic = "Referee request"
                        Request_with_reminder.Requestee.Title = "Editor"

        PRECONDITIONS   Author.Referees NOT COMPLETE

        SATISFACTION    Author.Referees IS COMPLETE

        WITH            Editor   = Request_with_reminder.Requestee
                        Referees = Request_with_reminder.Reply


PROC    Referees_review_paper

        description     Asks the referees to review a paper and then gets their
                        reports.  If a referee cannot assist, obtains a
                        replacement referee from the editor.

        IS              [  Request_with_reminder-1 (2 weeks)
                         ' (Get_replacement_referee ¦ nil)
                         ' Request_with_reminder-2 (2 weeks)
                         ' Thank_referee ]*

        COND            Request_with_reminder-1(i).Requestee
                                = (Request_with_reminder-2(i).Requestee
                                                OR
                                   Get_replacement_referee(i).Referee)
                        Request_with_reminder-1.Topic = "Review request"
                        IF (Request_with_reminder-1(i).Reply = "No")
                          THEN (Get_replacement_referee(i).Referee NOT NIL)
                        Request_with_reminder-2.Topic = "Report request"
                        Request_with_reminder-2(i).Requestee
                                = Thank_referee(i).Referee

        PRECONDITIONS   Author.Referees IS COMPLETE

        SATISFACTION    Author.Reports IS COMPLETE

        WITH            Referee(i) = Request_with_reminder-2(i).Requestee
                        Report(i)  = Request_with_reminder-2(i).Reply
```

PROC Get_replacement_referee

description	Asks the editor for a new referee if a previously selected one is unable to review the paper.
IS	Get_referees_from_editor
COND	SIZE (Get_referees_from_editor.Referees) = 1
PRECONDITIONS	Author.Referee(i) NOT AVAILABLE
SATISFACTION	Author.Referee(i) IS AVAILABLE
WITH	Referee = Get_referees_from_editor.Referees

'ROC Get_decision_from_editor

description	Asks the editor to accept or reject a paper after all the reviews are in.
IS	Request_with_reminder (2 weeks)
COND	Request_with_reminder.Topic = "Decision" Request_with_reminder.Requestee.Title = "Editor"
PRECONDITIONS	Author.Reviews IS COMPLETE
SATISFACTION	Author.Decision NOT NIL
WITH	Editor = Request_with_reminder.Requestee Decision = Request_with_reminder.Reply

ROC Inform_author

description	Tells the author if the paper was accepted or rejected.
IS	Tell_information
COND	Tell_information.Topic = "Decision"
PRECONDITIONS	Author.Decision NOT NIL
SATISFACTION	Author.Informed IS TRUE
WITH	Author = Tell_information.Informed Decision = Tell_information.Info

PROC Inform_executive_editor

description	Tells the executive editor if the paper was accepted or rejected.
IS	Tell_information
COND	Tell_information.Topic = "Decision" Tell_information.Informed.Title = "Executive Editor"
PRECONDITIONS	Author.Decision NOT NIL
SATISFACTION	Author.Exec.Informed IS TRUE
WITH	Exec = Tell_information.Informed Decision = Tell_information.Info

PROC Thank_referee

description	Thanks the referee for reviewing the paper.
IS	Send_a_message
COND	Send_a_message.Topic = "Thanks"
PRECONDITIONS	Author.Referee(i).Review NOT NIL
SATISFACTION	Author.Referee(i).Thanked IS TRUE
WITH	Referee = Send_a_message.Recipient

PROC Author_withdraws

description	Recieves notification from the author that s/he wishes to withdraw a paper.
IS	Be_told_information
COND	Be_told_information.Topic = "Withdrawal"
PRECONDITIONS	Author.Paper NOT NIL
SATISFACTION	Author.Withdrew IS TRUE
WITH	Author = Be_told_information.Informer

APPENDIX C

MEETING ARRANGMENT PROCEDURES

PROC Get_user_specification

description	Obtains a collection of information from the user.
IS	Obtain_information ' Update_information* ' Task_complete
COND	Obtain_information.Topic = FOR ALL i: Update_information(i).Topic
PRECONDITIONS	Info IS NIL
SATISFACTION	Info IS COMPLETE
WITH	Topic = Obtain_information.Topic Info = UNION (Obtain_information.Info, FOR ALL i: Update_information(i).Info)

PROC Get_attendees

description	Obtains the list of attendees desired at a meeting.
IS	Get_user_specification
COND	Get_user_specification.Topic = "Attendees"
PRECONDITIONS	-
SATISFACTION	Attendees IS COMPLETE
WITH	Attendees = Get_user_specification.Info

```
PROC    Get_rooms

        description     Obtains the list of possible rooms for a meeting.

        IS              Get_user_specification

        COND            Get_user_specification.Topic = "Rooms"

        PRECONDITIONS   -

        SATISFACTION    Rooms IS COMPLETE

        WITH            Rooms = Get_user_specification.Info

PROC    Get_times

        description     Obtains the list of possible time slots for a meeting.

        IS              Get_user_specification

        COND            Get_user_specification.Topic = "Times"

        PRECONDITIONS   -

        SATISFACTION    Times IS COMPLETE

        WITH            Times = Get_user_specification.Info

PROC    Get_attendee_availablity

        description     Finds out when the attendees are available.

        IS              Select_choices*

        COND            Select_choices.Topic = "Avaliablity"
                        FOR ALL i: SAME (Select_choices.Query)

        PRECONDITIONS   (Meeting.Times NOT NIL) AND (Meeting.Attendees NOT NIL)

        SATISFACTION    Attendee.Availability IS COMPLETE

        WITH            Attendee(i)  = Select_choices(i).Requestee
                        Attendees    = UNION (FOR ALL i: Attendee(i))
                        Time_slots   = Select_choices.Query
                        Free_time(i) = Select_choices(i).Reply
                        Free_times   = UNION (FOR ALL i: Free_time(i))
```

```
PROC    Modify_choices

        description     Has the user update her/his choices of attendees and/or
                        time-slots until at least one common time is found.

        IS              (Update_information+ ' Get_attendee_availability)*

        COND            Update_information.Topic = "Times" OR "Attendees"
                        IF Update_information(i).Topic = "Times"
                          THEN Get_attendee_availablity(i).Time_slots
                                 = Update_information(i).Info
                        IF Update_information(i).Topic = "Attendees"
                          THEN Get_attendee_availablity(i).Attendees
                                 = Update_information(i).Info

        PRECONDITIONS   (Meeting.Times NOT NIL) AND (Meeting.Attendees NOT NIL)

        SATISFACTION    Meeting.Common-times NOT NIL

        WITH            Attendees    = Get_attendee_availability.Attendees
                        Common-times = INTERSECTION
                                         (Get_attendee_availability.Free-times)

PROC    Select_room

        description     Chooses a suitable (and available) room and time.

        IS              Select_choices

        COND            Select_choices.Topic "Room"

        PRECONDITIONS   (Meeting.Rooms NOT NIL) AND
                         (Meeting.Common-times NOT NIL)

        SATISFACTION    Meeting.Selected-room NOT NIL

        WITH            Possible-room/times = Select_choices.Query
                        Selected-room       = Select_choices.Reply.Room
                        Selected-time       = Select_choices.Reply.Time
```

```
PROC    Check_with_attendees

        description     Verifies that the attendees can attend a meeting
                        at the selected time and place.

        IS              (Request_with_reminder (1 day))*

        COND            FOR ALL i: Request_with_reminder.Topic = "Verify"
                                   SAME (Request_with_reminder.Query)

        PRECONDITIONS   Meeting.Selected-room NOT NIL

        SATISFACTION    Meeting.Attendees.Verified COMPLETE

        WITH            Attendees = UNION
                                    (FOR ALL i: Request_with_reminder.Requestee)
                        Room      = Request_with_reminder.Query.Room
                        Time      = Request_with_reminder.Query.Time
                        Verify(i) = Request_with_reminder(i).Reply
                        Success   = LOGICAL-AND (FOR ALL i: Verify(i))

PROC    Confirm_room

        description     Confirms the reservation of the selected room and time.

        IS              Tell_information

        COND            Tell_information.Topic = "Confirm"
                        Tell_information.Informed.Title = "Room scheduler"

        PRECONDITIONS   Meeting.Attendees.Verified IS SUCCESSFUL

        SATISFACTION    Meeting.Selected-room.Confirmed IS TRUE

        WITH            Room = Tell_information.Info.Room
                        Time = Tell_information.Info.Time

PROC    Confirm_attendees

        description     Sends final notice to the attendees of the meeting
                        time and place.

        IS              Tell_information*

        COND            FOR ALL i: Tell_information(i).Topic = "Final Notice"
                                   SAME (Tell_information(i).Info)

        PRECONDITIONS   Meeting.Attendees.Verified IS SUCCESSFUL

        SATISFACTION    Meeting.Attendees.Confirmed IS TRUE

        WITH            Attendees = UNION (Tell_information.Informed)
                        Room      = Tell_information.Info.Room
                        Time      = Tell_information.Info.Time
```