KNOWLEDGE-BASED OFFICE AUTOMATION AND CSCW

Michel TUENI, Jianzhong LI

Advanced Studies Department / Bull
7, rue Ampère
91343 Massy Cedex / France

Abstract

The cooperative aspect of office work entails developing OIS systems which support office tasks that have to be executed in a collaborative manner. In particular, many systems have been developed using speech act elements as the theoretical underpinnings. Instead of using speech act elements, we use operators (e.g., send, request and acknowledge) as basic building blocks. We have developed a knowledge-based system called AMS (Activity Management System) that is specifically tailored to support CSCW. The system embodies the syntax and operational semantics of the various office operators. These operators can then be concatenated in the same manner as speech act elements to support CSCW.

1 - Introduction

There has been a proliferation of office support products, such as Multimedia Editors, Data Base Management Systems, Spreadsheets and Electronic mail. These tools have considerably alleviated the office workers in their tedious day-to-day work and increased their productivity. As the computer becomes more and more advanced, sophisticated and easier to use, its presence in the office environment becomes more pervasive. Consequently, computer aided office support products become more and more "intelligent" and diversified. There is now a full array of such products : document production tools, basic communications tools such as Electronic Mail, decision-making aid systems, complex systems such as those supporting collaborative efforts within working groups, etc. Nevertheless, the emphasis always is the same: making office workers more productive and efficient. These systems fall under "Office Information System" (OIS).

Many general OIS systems, based on different methodologies and approaches, have been proposed. These OIS systems seek to support different levels of office tasks in an integrated way with the same emphasis: delegating as much office work to the computer as possible. A good review of the existing systems can be found in [Bracchi, Pernici 84].

Recently, the cooperative aspect of office work has become an intensive area of research. This is due to the intrinsic nature of office work coupled with the advances made in the area of computer technology. Many systems have been designed using Speech Act theory [Searle 69] (with a background in linguistics). The theory categorizes all sentences into classes relevant to their functions in interpersonal relationships: assertions, directions, commitments, declarations and expressions. Examples of such systems are XCP [Sluzier, Cashman 84], Coordinator [Winograd, Flores 86], CHAOS [De Cindio 86], SAMPO [Lyytinen 87] and COSMOS [Wilbur, Young 88]. These systems have been developed specifically to support CSCW.

Ongoing research in Artificial Intelligence (AI) also provides a rich repertoire of concepts for knowledge engineers to develop knowledge-based OIS to support CSCW. Artificial intelligence formalisms have been used to embed knowledge of the organization in order to help the office worker in his problem-solving tasks. Many systems using this approach have been developed. POLYMER [Croft 88] is a good example.

We take the view that a synthesis of concepts from the Speech Act theory and Artificial Intelligence would provide a more rigorous, comprehensive theoretical foundation for developing a well-integrated OIS to support CSCW. This, then, is the objective of this paper.

In the next section, we will highlight the collaborative nature of office work. In section 3, we will present the Activity Manager System, a knowledge representation formalism which allows knowledge representation and organization of office tasks. In section 4, we will describe the CSCW model based on the AMS representation formalism.

2 - Collaborative Nature of Office Tasks

Most existing office models (e.g., OSSAD [88], OPAL [Ahlsen et al. 84]) focus primarily on representing office procedures without explicating the collaborative nature of office work or what Auramaki et al [88] term "social features of offices." Offices, then, are systems of communicative actions". Through these communicative actions, office workers undertake to perform actions by making commitments. In fact, De Michelis et al [88] go as far as to characterize "human cooperation as a starting point towards the characterization of organizational systems."

An apparently simple example will demonstrate the cooperative facet of office work. Consider, for instance, a commercial department where several employees want to take business trips. First, each of these employees would have to consult with his/her immediate superior. His/her immediate superior will have to ensure that the employee's absence will not affect the ongoing work of the department. If the employee encounters no objections from his/her immediate superior, then he/she will have to follow some prescribed administrative procedure that reflects the rules and the constraints of the organization. Having done that, authorization has to be given by the director of the division of which the commercial department is a subunit. The director has to determine if the objectives of the trip serve to strengthen the long-term goal(s) of the organization. After the approval is granted, the employee will have to liaise with the Accounting Department to fill in a form to get a ticket from the travel agency, and to fill in another form to get an advance of funds.

In the above example more than two departments are involved. The need for cooperation is augmented by the fact that several individuals from each department may be involved. These individuals have their own sets of knowledge, beliefs and expectations. In order for such an apparently "highly-structured" task group to be successfully executed, an effective network of human interactions and cooperation must exist as a sine qua non. This network acts as a synchronization mechanism, ensuring that the various actions are performed in a concerted manner.

An OIS that is aimed at supporting CSCW should incorporate the following characteristics:

1) It should permit individuals to model and generate their own work modules. This can range from the simple act of copying a document to the cognitively taxing act of writing an article;
2) This is an extension of 1). Having created the information objects, be they simple or complex, the individuals may want others to modify, edit or revise these objects. As such, the OIS should be able to support migration of information objects in the classical office manner as documents are moved from one desk to another; and
3) It should provide a rigorous formalism that will function as a structuring mechanism. Information objects created by A and B may be needed by C simultaneously if C is to able to carry out his/her part of the cooperative work process. This entails

synchronizing the arrival of A's and B's information objects. Other scenarios exist which require the formalism to impose temporal-order sequencing. A possible scenario is one where A, B, C and D are engaged in information pipelining, which is a form of cooperative work. A more complex scenario in which both temporal-order constraints and synchronization are essential ingredients is that of assisting a manager in scheduling his team. The scenario involves "scheduling meetings, monitoring the progress of subgoals, altering the managers to deadlines, and real-time scheduling" [Goldstein, Roberts, 1979]. Implicit in this scenario is the need to handle missing information inasmuch as uncertainty pervades many facets of office work.

3 - Activity Manager System (AMS)

We intend to use AMS as the underlying structuring mechanism for CSCW. AMS is a knowledge-based system which supports the representation and execution of procedural knowledge. The procedural paradigm suits the representation of office knowledge, and this has dictated the design parameters of the system. In this section, we will present the underlying knowledge representation concepts of the AMS in a general purpose terminology. Issues related to office information system and graphical representation problems can be found respectively in [Tueni et al. 88].

3.1 - Introduction to AMS AI formalism

Procedural knowledge describes sequences of things to do or goals to be achieved. The procedural approach of representing knowledge is used in various domains such as office task management, natural language understanding, project management, factory scheduling and, in particular, planning.

Earlier knowledge-based systems, such as MYCIN [Shortliffe 76], represent procedural knowledge by using rule formalisms. Here, rules represent independent declarative pieces of knowledge, and this allows knowledge base to be modified easily. However, the sequential nature of procedural knowledge renders its representation by rules problematic. As Georgeff & Bonollo [83] aptly put it :"Because of the homogeneity of the rule representation, it is not possible to distinguish between those rules for which the order of invocation is important and those for which it is not."

Georgeff et al. [85] considered a plan generated by a planner as an external behavior of intelligent agents, and proposed the notion of process for representing procedural knowledge. A *process* consists of both a purpose description called the *invocation* part and a body represented in the form of network where the nodes (i.e. control points) are labelled by state conditions and arcs are labelled by goals. It is the body which describes the sequence. During the satisfaction process of an abstract goal, other *processes* may be invoked to refine the goal into a series of more specific goals. But this approach is rather "flat" in the sense that no effort is made to represent abstraction and generalization within the network in order to obtain an explicit hierarchical representation. This is one of the issues which we will address.

Friedland & Iwasaki [85] use the concept of a skeletal plan, which is actually a schema to represent plans at varying levels of abstraction. This hierarchical representation gives the possibility to refine the steps of an abstract plan by finding out the more specific plans and so on, as well as the possibility to have several plans (with no relation among them) matching the same goal at different levels of abstraction. But "there is often a choice to be made among picking a very specific plan that will require little refinement work or picking a more general plan" [Friedland, Iwasaki 85]. One wrinkle about the skeletal plan is that there exists redundant plans at different levels of abstraction. One way to circumvent this is to use links to reference each other. Not only is this more effective, but it is also a way in which the issue of reusability can be introduced implicitly.

Schank [82] made a significant contribution by suggesting that the script-like structures do not exist as permanent memory structures; they would have to be constructed from the higher-level general structures as needed by consulting the rules governing the particular situation. Schank also proposed the so-called MOPs (Memory Organization Packets) model to address knowledge organization issue. He stated that "each time a high-level knowledge structure is accessed during normal processing, the piece of the story being processed relevant to that structure is stored at that processing-related node".

Most hierarchical representation mechanisms are used mainly to reduce the searching space, and thus the complexity of the problem [Wilkins 86]. Not many of them have abstract structures for organizing procedural knowledge as MOPs do.

In this section, we will propose a model for hierarchical representation of procedural knowledge which emphasizes:

1) Describing procedure at a reasonable degree of complexity and completeness.
2) Representing procedures at different levels of abstraction.
3) Organizing hierarchies by means of abstract entities.
4) Reusability of the knowledge at different levels of abstraction.

The model has been developed within the context of office procedures, but it can be considered as a general purpose paradigm. This paradigm can also be used to represent procedural knowledge in planning.

3.2- Procedural knowledge representation

Procedural knowledge is often expressed in the form of *plans* or *networks* which correspond to sequences of steps. Graphical representation has an inherent elegance in capturing procedural sequences explicitly even when these sequences exhibit phenomena such as concurrency, conflict and synchronization. We use a network paradigm together with operators to depict any procedural sequences. The operators are as follows:

1) The *loop* for expressing recurrence.
2) The *and-branches* and *rendez-vous-points* for expressing the parallelism among sequences.
3) The *or-branches* for expressing alternatives among sequences.

3.3 - Basic Concepts of AMS Formalism

We will explicate the fundamental concepts of the AMS formalism, namely, activity, state, action, network and MOPA.

a - Activity

The *Activity* concept is the basic entity of the AMS. Everything we do may be considered an activity, regardless of the abstraction level at which it is situated. Before performing an activity in some manner, we should check if the necessary preconditions are satisfied.

Three types of information are encapsulated by the activity concept:

- A Start-State, describing the information to be checked before performing the activity;
- A Caused-State, describing the effect (or the reached goal) caused by the activity execution;
- The Body that describes the way the activity will be performed.

The Body represents the knowledge used to perform the activity. We can distinguish two types of activities: (1) terminal activities whose bodies are terminal actions not capable of being decomposed further, and (2) complex activities whose bodies are sequential descriptions capable of being decomposed further. In the context of our discussion, the body will be represented by an Activity-network or a MOPA (see next section).

An activity might be regarded as a rule-like entity if one takes its start-state as the precondition, its caused-state as the then-part and its body as the action-part. But our activity concept is more general than rules in that it can be situated at any level of abstraction. In our hierarchical representation, it provides us with uniformity of representation at any level of abstraction. This means at given level, the activities are conceptually the same.

The *purpose* of an activity describes the goal, and corresponds to the invocation part of the activity. The purpose can be viewed as a pattern that is used to retrieve the activity by matching that pattern with a request (see below - Node definition).

b- State

The state is the key concept of AMS. It represents the information that should be checked before performing a task (Start-State), or the effect of the execution of a given task (Caused-State). When evaluated, a state returns a boolean appreciation of the application domain. States are designed to allow the knowledge engineer to represent explicitly those important facts that are significant in a domain and to track their evolution during execution.

c- Action

Action is the concept which embodies a primitive function (e.g., a lisp function). The activity which contains such a *body* is called a *terminal activity* . The purpose of the action concept is to interface the AMS with the real application domain.

d- Activity-Network (AN)

An *Activity-Network* consists of a set of nodes and a set of directed edges between these nodes. The nodes represent the location where activities are applied, and the edges represent the precedence relations between the nodes. Two operators have been attached to nodes to handle the evaluation and synchronization of nodes. The input operator (*OR / AND*) handles the input edges, while the output operator (*OR / AND*) handles the output edges. The following figure shows how these operators work.

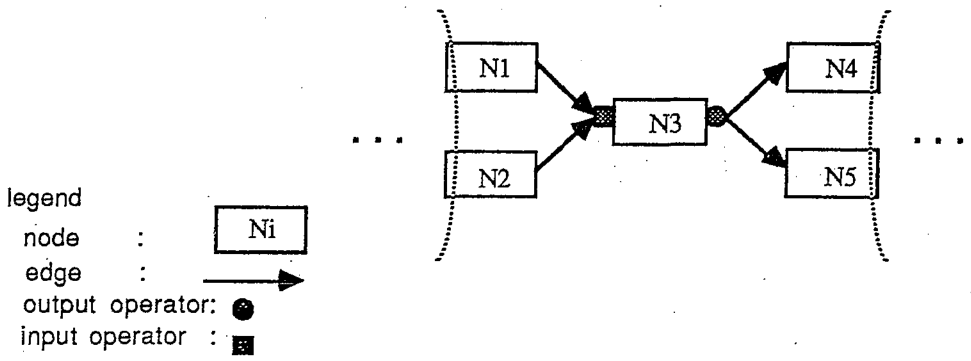


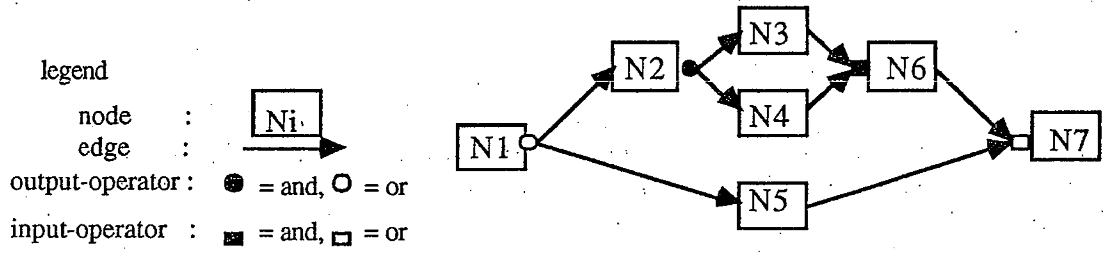
Figure 2 - Network

It is worth mentioning that, from the control structure point of view, an activity-network behaves as a Petri-net [Peterson 77]. The logical operators OR and AND depict conflict, synchronization and concurrency.

What distinguishes our network representation from other representation formalisms (e.g., Petri nets) is its ability to represent abstractions. Two additional slots are attached to a node, the *request* slot and the *has-activity* slot. The activity which satisfies the request may be known directly or not by a node. In the former case, the activity is directly pointed out by the *has-activity* slot. The matching serves in this case as a context passing mechanism. In the later case, the corresponding activity is dynamically retrieved from the ability list. Such a node is termed an *abstract node* and an AN which contains an abstract node is termed *abstract activity-network.*

As pointed out in the introduction, office tasks and administrative procedures require the declarative representation of knowledge and the representation of sequences of things to do. This sequence is represented by an activity network.

Figure 3 shows an example of a simple AN. We start by an evaluation of the node N1. The "or output operator" means that either N2 or N5 should be evaluated (not both of them), the "and output operator" attached to N2 states that N3 and N4 should be evaluated. When reaching N6 the "and input operator" indicates that N6 waits until the N3 and N4 are evaluated before being considered. The node N7 is evaluated when N6 or N5 has been evaluated.


legend
node :
Ni
edge :
output-operator : ● = and, ○ = or
input-operator : ■ = and, □ = or
N1
N2
N3
N4
N5
N6
N7
Figure 3 - An Activity Network

e- Memory Organization Packet for Activities (MOPA)

A requirement of hierarchical representation is the ability to reason at different levels of abstraction, and to extract abstract knowledge that is reusable in various situations. Knowledge reusability can be achieved by organizing knowledge in abstract entities. Abstract knowledge is used to organize concrete knowledge and construct ANs at different levels.

We introduce a concept of MOPA (Memory Organization Packet for Activities) as a knowledge organization mechanism and a knowledge processing. Here, we share the same theoretical basis as [Schank 82].

MOPAs exist at different levels of abstraction. A MOPA (at any level) has two pointers: the *concrete-use-of* pointer and the *abilities* pointer. The concrete-use-of pointer points to another MOPA at one level up in the abstraction hierarchy. For instance, the MOPA "Cooperation-request" points to the MOPA "Negotiation", the former having a greater degree of specificity than the latter (see Figure 4). The abilities pointer points to a list of activities associated with the MOPA.

Given any context, we can dynamically generate ANs associated with their corresponding MOPAs only if we start with a general AN as the upper bound, and a MOPA as the lower bound. The nodes of the general AN (i.e., the upper bound) can either be abstract nodes, or concrete nodes. When there are abstract nodes in the general AN, the matching mechanism tries to find the corresponding activities whose *purposes* match with the *requests* of the abstract nodes. Once the matching succeeds, links are established and the AN associated with MOPA at one level down the hierarchy is generated.

The advantages of this model of representation are summarized as follows:

1) As in rule-based systems, the control is separated from the knowledge, and programming is replaced by the explicit declarative representation of packets of knowledge. How to use these packets is left to the system.

2) An explicit representation of procedural knowledge by means of the *Activity-network* concept which allows us to describe any procedural situation with reasonable ease.

3) A general representation of the rule-like entity (as used in classical expert systems) by means of the *Activity* concept.

4) A hierarchical representation which allows the dynamic generation of ANs at varying levels of abstraction. The MOPA concept plays the role of organizing the knowledge in an appropriate manner.

5) A uniform representation which allows us to have the same view at any level of abstraction. Indeed, the activities and activity-networks which are used for describing each level of procedural knowledge are conceptually the same.

The simple example discussed in Section 2 (business trip) is illustrative of a typical office procedure involving several actors. Such a procedure can be implemented using AMS formalism.

Many CSCW applications have been designed with the goal of supporting commitments, namely, helping office workers by permitting them to define their commitments explicitly. One of the areas which deals with this issue is the Speech Act theory.

4 - A Speech Act Model on top of the AMS

We use reusable operators (e.g, send, request, acknowledge and answer) as basic building blocks (built on AMS formalism) instead of speech act elements. The conceptual basis is the same because these operators perform the same function as speech act elements.

The architecture of AMS can be stratified into three layers. The kernel of the system includes the following fundamental features:

1) A control structure to handle the interruption, resumption and the cancellation of tasks;

2) A mechanism to schedule tasks that are in progress;

3) A mechanism to handle missing information since uncertainty is a basic phenomenon in office work.

On top of the kernel is the knowledge representation stratum which encapsulates abstract knowledge of the modeled domain. What is interesting about the AMS formalism is that the abstract knowledge can be rendered concrete once specific parametric values are given, thus effectively capturing the notion of reusability. No programming is required. All that is required it to concretize abstract packets of knowledge through explicit declarations.

The topmost stratum is the application domain which embodies the syntax and the operational semantics of the various office operators such as create, send, copy, request, answer and so on. Within the context of CSCW, these operators serve useful functions in that they can be concatenated in the same manner as speech act elements to support conversations for action and conversations for clarification. A complete description of the architecture of AMS is beyond the scope of this paper.

The AMS formalism is more in line with the approach suggested by [Cohen 78] in that it (AMS) offers a planning system which allows the representation of activities and networks (operators) at different levels of abstraction. Speech act elements can thus be described by the AMS concepts.

We will provide an example to illustrate how speech act elements or what we call office operators can be concatenated to depict the network of interpersonal relationships that exist in the office. We will also demonstrate these office operators can be reused in different situations.

In an office, exchanging information is done through conversations, involving any number of people. A conversation follows a specific scheme which is sequence of speech acts.

Figure 4 depicts an example of how the reusable operators of AMS can be concatenated to model a "request for cooperation." The modeling can be divided into three levels. The first level describes how a conversation is initiated and this can be reused in any type of conversation, provided that a message is sent to somebody designated to handle the response. The second level sequences the negotiation phase by defining it in a more specific way how the response is to be handled. It can also be reused in different situations where negotiation is foreseen. The third level specifies the activities that have to be executed for cooperation among agents to take place. There are three ways in which cooperation is handled: accepting, denying or making a counter-proposal. The "answer for cooperation" activity can be defined in the same way.

The comments in the boxes (figure 4) are patterns which stand for either a purpose in the case of an activity, or a request in the case of a node. The shaded boxes are nodes which point explicitly to an activity, while the plain ones are abstract nodes.

At the most abstract level (level-1 in figure), we have the *conversation* abstract AN which starts a conversation. At this level, two nodes, "prepare the @message" node and "handle <response" node, are left abstract (plain). That means at these places we know what must be done, but not how to do it. It depends on the context of the conversation.

The *negotiation* MOPA (level-2) adds additional knowledge to the context (pointed out by the *abilities* slot) - "handle >response" activity which will be matched with the "handle <response" node. By reusing the *conversation* AN (pointed out by *concrete-use-of* slot), a more specific AN is generated at this level. Two nodes are still abstract: they will be filled in when this AN is used in a concrete case of *negotiation*.

The same reasoning mechanism can be applied to the *cooperation request* MOPA (level-3). Note that, at this level, three activities might be matched with "handle >request for <status" node. It depends on whether the response is *accepted* or *counter-proposed* or *denied*.

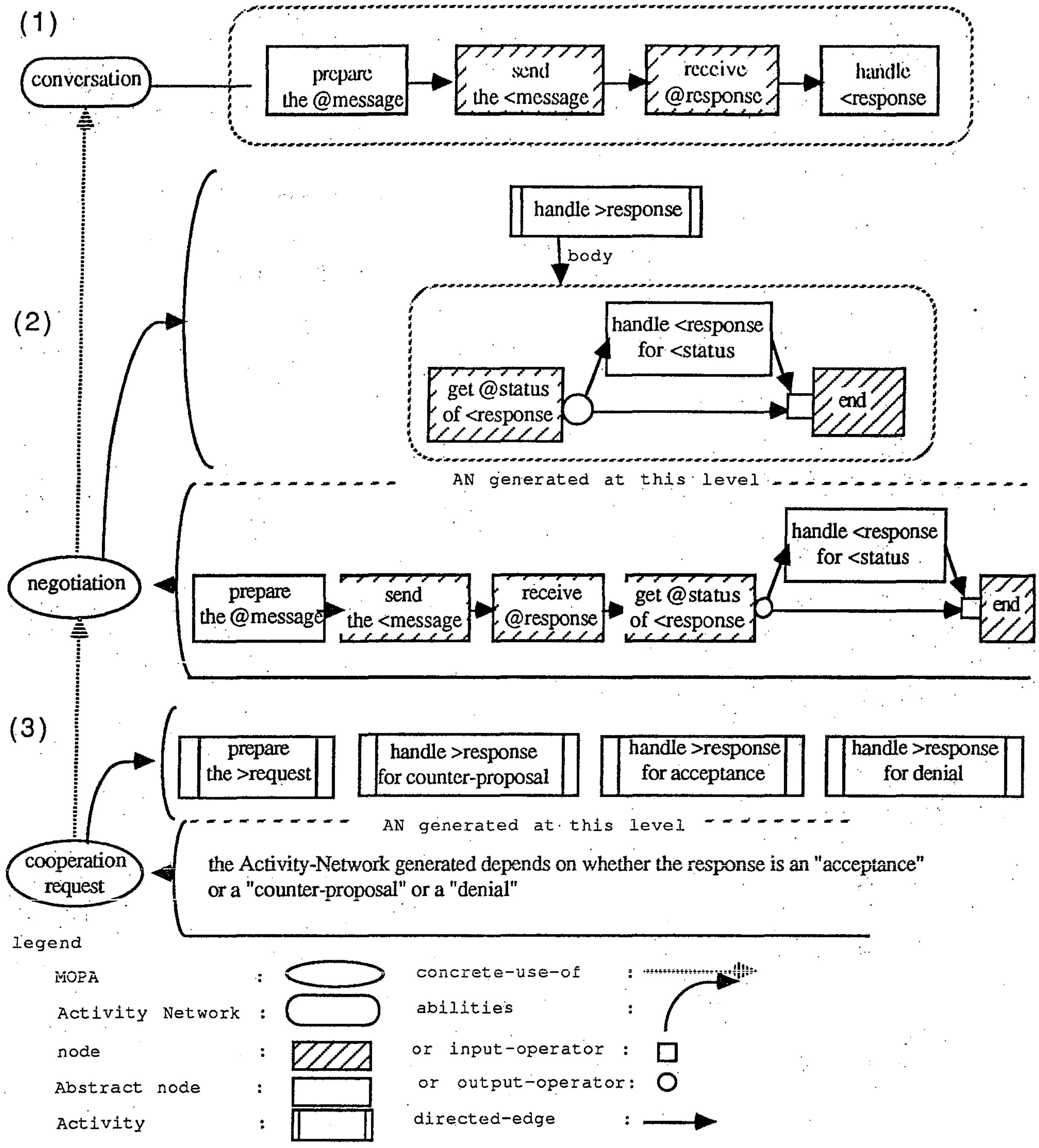


Figure 4 - request a cooperation

5 - Conclusion

Our objective is the design of a general knowledge-based system for supporting the CSCW. We stressed the collaborative nature of office work, and argued that an effective OIS should be based on a firm theoretical basis if it is to provide sound support for CSCW.

The AMS formalism can provide the theoretical underpinnings for the development an effective OIS. Five main points characterize the AMS representation formalism:

1) As in rule-based systems, the control is separated from the knowledge, and programming is replaced by the explicit and the declarative representation of packets of knowledge. How to use these packets is left to the system.

2) An explicit representation of procedural knowledge by means of the *Activity-network* concept which allows us to describe any procedural situation with reasonable ease.

3) A general representation of a rule-like entity using the *Activity* concept.

4) A hierarchical representation which allows the dynamic generation of ANs at each level of abstraction. The MOPA concept plays the role of organizing the knowledge in an appropriate manner.

5) Uniformity of representation which allows us to have the same view at any level of abstraction. Indeed, the activities and activity-networks which are used to describe each level of procedural knowledge are conceptually the same.

The CSCW module models negotiations and commitments between different agents. Using the Speech Act theory as the conceptual base, the CSCW module has been designed to support non-linear planning. Thus, the communication process encased in the CSCW module allows agents to exchange information and planners to retain consistency and to synchronize distributed nodes of activities.

References

[Ahlsen et al. 84] M. ANLSEN, A. BJÖRNERSTEDT, S. BRITTS, C. HULTEN and L. SÖDERLUND, " An Architecture For Object Management In OIS", ACN Transactions On Office Information System 2(3), July 1984;

[Auramaki et al. 88] E. AURAMÄKI, E. LEHTINEN and K. LYYTINEN, "A Speech-Act--Based Office Modeling Approach", ACM Transactions on Office Information Systems, Vol. 6 N° 2, April 1988;

[Bracchi, Pernici 84] G. BRACCHI and B. PERNICI, "The Design Requirements of Office Systems", ACM Transaction on Office Information Systems, Vol. 2, n°2, April 1984;

[Cohen 78] P. R. COHEN, "On knowing what to say: Planning speech acts", Ph.D. Thesis, Technical Report N°118, Department of Computer Science, University of Toronto, January 1978;

[Croft et al. 88] W.B. CROFT, LEFKOWITZ, U. MASS "Using a planner to support office work", In proceedings Conference on Office Information Systems, Palo Alto, March 1988;

[De Cindio 86] F. De Cindio, G. De Michelis, C. Simone, R. Vassalo et A. Zanaboni, "Chaos as coordination technology", CSCW '86 proc. of the Conference on Computer-Supported Collaborative Work, Austin, Texas 1986;

[De Michelis et al. 88] G. De Michelis, F. De Cindio and C. Simone, "Groups In A Language/Action Perspective", Technical report, Università degli Studi di Milano, 1988;

[Flores et al. 88] F. FLORES, M. GRAVES, B. HARTFIELD and T. WINOGRAD, "Computer Systems and the Design of Organizational Interaction", ACM Transactions on Office Information Systems, Vol.6 N°2, April 1988;

[Friedland, Iwasaki 85] P. E. FRIEDLAND and Y. IWASAKI, "The Concept and Implementation of Skeletal Plans", Report N° KSL 85-6, Computer science department, Stanford University, California, 1985;

[Georgeff, Bonollo 83] M.P. GEORGEFF, U. BONOLLO, "Procedural Expert Systems", Proceeding 8th IJCAI, Karlsruhe, FRG, 1983;

[Georgeff et al. 85] M.P. GEORGEFF, A.L. LANSKY, and P. BESSIERE, "A Procedural Logic", In Proceeding 9th IJCAI, Vol. 1, August 1985, Los Angeles, California;

[Goldstein, Roberts 79] I. P. GOLDSTEIN, B. ROBERTS, "Using Frames In Scheduling", In (Ed.) Winston P. H. & Brown R. H., Artificial Intelligence: An MIT Perspective, Vol. I 1979;

[Holt 85] A.W. HOLT, "Coordination technology and Petri Net", In (Ed.) G. Rozenberg, Advances in Petri Nets 1985, Lecture Notes In Computer Science 222, 1985;

[Koo 88] Charles C. Koo, "A Commitment-based Communication Model For Distributed Office Environments", ACM Transaction on Office Information Systems,1988;

[Lyytinen 87] LYYTINEN, "Different perspectives on information systems" in ACM Comp. Surv. 19, 5-46, 1987;

[OSSAD 88] In (Ed.) De Antonellis V., Bestmüler E., Calmes F., Charbonnel G., Conrath D. W., Dumas P. De Petra G., De Sanctis C., Simone C. and Sorg S., "Office System System Analysis and Design", OSSAD manual, January 1988;

[Peterson 77] J.L. PETERSON, "Petri Nets", ACM computing Surveys, 9, N°3, September 1977;

[Schank 82] R.C. SCHANK, "Reminding and memory organization : An introduction to MOPS", in : W. Lehnert et M. Ringle (Eds.), Strategies for Natural Language Processing (Lawrence Erlbaunm, Hillsdale, NJ, 1982);

[Searle 69] J.R. Searle, "Speech Acts: An essay in the philosophy of language", Cambridge University Press, Cambridge, 1969;

[Shortliffe 76] E.H. SHORTLIFFE, "Computer Base Medical Consultation: MYCIN", American Elsevier, New York, 1976;

[Sluzier, Cashman 84] S. SLUZIER, P.M. CASHMAN, "XCP: An experimental tool for supporting office procedures", IEEE 1984 Proceedings of the first International Conference on Office Automation, Silver Spring, MD:IEEE Computer Society, 1984;

[Tueni et al. 88], M. Tueni, J. Li, and P. Fares, "AMS: A Knowledge based approach to tasks representation manipulation and organization", COIS-88, Palo Alto, March 1988;

[Wilbur, Young 88] "The Cosmos Project - A Multi-disciplinary Approach to Design for Computer-supported Group Working", Euteco '88 proc. of Research into Networks and Distributed Applications, April 1988;

[Wilkins 86] D. E. WILKINS, "Hierarchical Planning: Definition and Implementation", Proceedings 7th ECAI, July 1986;

[Winograd 86] Terry Winograd, "Language perspective on the design of cooperative work", CSCW '86 proc. of the Conference on Computer-Supported Collaborative Work, Austin, Texas 1986;

[Winograd, Flores 86] T. Winograd, F. FLORES, "Understanding Computer and Cognition: A New Foundation for Design", Norwood, New Jersey: Albex, 1986 and Addison-Welsey, Reading, Mass., 1987;

[Zisman 78] M.D. ZISMAN, "Use of Production Systems for Modeling Asynchronous, Concurrent Processes", In Pattern-Directed Inference Systems, Waterman and Hayes-Roth, Eds., Academic Press, New-York, 1978;