Considering the User in Mixed-Initiative Meeting Management

A. Cesta $^{\pm}$ and D. D'Aloisi $^{\mp}$ and R. Brancaleoni $^{\mp}$

$^{\pm}$ IP-CNR, National Research Council [*]
Viale Marx 15, I-00137, Rome, Italy
amedeo@pscs2.irmkant.rm.cnr.it

$^{\mp}$ Fondazione Ugo Bordoni [†]
Via B. Castiglione 59, I-00142 Rome, Italy
{dany,gheo}@fub.it

October 7, 1996

Abstract

This paper describes a multi-agent system able to manage the meeting schedule of a set of users. The problem of meeting scheduling has been considered because represents an example of how routine daily activities may be delegated to software agent to relief activity overloading of human agents. In the paper a general agent architecture is described which is used to realize different kinds of agent in a specific system called MASMA (Multi Agent System for Meeting Automation). The way MASMA addresses various aspects of the agenda management problem is described and, in particular, several issues concerning the acceptability of the agent approach by human users are discussed. To increase acceptability, an important aspect is the attention paid

[*]Partially supported by Modelage and SARI.

[†]Work carried out in the framework of the agreement between the FUB and the Italian PT Admnistration.

to the problem of user control over agents activities. The possibility of task delegation is considered a relevant achievement of software agent technology, but issues like, non-invasion, possibility of inspection, and privacy should be taken into account. In the paper the way these problems are dealt with in MASMA is also described.

1 Introduction

Large computer networks—like Internet—allow users to access fast huge amount of information and data but also caused them a cognitive overloading due to the need of acquiring and learning the use of new tools. The network could help to mechanize the resolution of problems previously fulfilled *manually*: nevertheless it is quite difficult to design really useful and supporting tools without further increasing the user's involvement.
The wide use and success of tools solving problems connected to the network, e.g., e-mail managing, information filtering, personal agenda scheduling, are strictly depended on their conjunction with effective and supportive interfaces: such interfaces should be cooperative with respect to the users and competent with respect to the task they are supposed to perform. Moreover the interfaces should adapt themselves to the user's needs by relieving him from learning the use of tools and from performing repetitive patterns of actions. The interface should not only be a filter towards applications, but it should perform additional activities besides those strictly requested to enhance the offered performance: it should accompany the growth of the user's competence to satisfy his needs even before explicit requests. That can be accomplished by accounting for the user's profile and behavior.
Given the requirements and the features described these interfaces acts as *autonomous agents* able to actively propose solutions and more generally to substitute the user in the most repetitive activities behaving as *personal assistants* [2].
The developing of supporting agents is particularly relevant in tasks and operations that are time-consuming and routine. An interesting problem that an agent-base system can dealt with is the automated scheduling of personal agendas. Its resolution—usually performed "by hand"—can be improved by a mechanized process and by the existence of the network. Moreover it is an emblematic problem to show some aspects of the agent-user relationship: the need of non-invasion and privacy, the possibility of inspecting what the agent is doing, the increasing level of user's confidence in agent's competence, the agent's capability of learning by user's

behavior.
This paper describes MASMA (Multi-Agent System for Meeting Automation), an agent-based and multi-agent system (MAS) designed to support users in managing their personal agenda and to organize and schedule appointments, meetings, small conferences and seminars. MASMA consists of a set of agents that cooperate among them each devoted to deal with a particular task or set of tasks. In developing MASMA the practical usability of MASs and their effectiveness in interacting with users assume a fundamental importance. The problem solved by MASMA is described in the next section. Then the architecture of the system and the classes of agents it involves are introduced. Each agent is an instance of a general architecture that is illustrated in the fourth section. Then a more detailed description of the main agent, the *Meeting Agent*, is supplied followed by a discussion about the focus of our research, the interaction with the user, and how MASMA deals with it. A conclusive section closes the paper.
It has to be pointed out that some issues and parts of the system can and must be improved (for instance the negotiation strategies and the optimality criteria in the solution search), but some fundamental ideas about the relevance of the user in agent-based system have been placed.

2 Automated Management of Meeting Agenda

Organizing meetings generally requires a massive organizational effort, complex negotiation strategies and a huge numbers of communication acts, e.g., e-mail messages, phone calls, faxes, etc. A significant reference problem is the so-called *secretaries' nightmare* [13] since generally are the secretaries who find a compromise among the different users' constraints, the availability of the resources and the need of satisfying their bosses. An efficient and optimal scheduling process could bring benefits in term of saving time and costs and of optimizing the information flow.
The problem can be seen as a particular application domain of distributed scheduling among a set of agents (Distributed Task Scheduling). In this context the tasks are the appointments to be fixed and the resources are the participants. To each resource (person) an agent is associated managing his agenda that organizes meetings by negotiating with other agents. It knows the user's preferences concerning the available dates, the information to be exchanged with the others, the agents to negotiate with, the user's interests and so on. Each meeting is organized by an agent, involves a number of other agents and is characterized by a set of constraints. The or-

ganizer agent leads the negotiation process proposing a set of possible time intervals, gathering the invitees' answers and figuring out a common solution. If necessary the process is repeated until an agreement is reached. The formal description of the problem is the following. Given a set A of Meeting Agents, each associated with a user, a meeting M_i is a tuple $M_i = \{I, h_i, d_i, T, C, < Place >, < Instrument >, AV\}$ where

- $I \subseteq A$ is a set of invitees consisting of a set $NI \subseteq A$ of necessary attendants and a set $OI \subseteq A$ of optional attendants such that $I = NI \cup OI$;
- h_i is the organizer agent or *host* of M_i;
- $d_i =< n_{day}, nh_1, \ldots, nh_{n_{day}} >$ is the duration of the meeting in which n_{day} is the duration in days and nh_{n_k} is the duration in hours for each day;
- $T = \{interval_1, \ldots, interval_n\}$ is the set of possible time intervals for the meeting. The generic interval is specified by the date plus its starting and end times, $interval_j =< day, t_{s_j}, t_{e_j} >$;
- C is the set of the possible sites where the meeting can take place;
- $< Place >$ is a set of *attribute-value* pairs that describes the structure hosting the meeting, i.e., type of building, capacity, etc.;
- $< Instrument >$ is a set of *attribute-value* pairs that describes the supporting instruments, i.e., microphone, VCR, etc.;
- AV describes the participants' availability with respect to the set of possible time intervals, i.e., $AV = \cup_{i \in I} AV_i$ where $AV_i = \{< interval_1, value_{i1} > \ldots < interval_n, value_{in} >\}$. $value_{ij}$ is a preference value in the set $\{high, medium, low, nil\}$.

All the users are connected to a network even if they can be located anywhere. A centralized management of the resources—in the sense of sites and instruments—is applied.

2.1 Related Works

Most of the systems developed in the past simply offer an interface toward the diary and a framework for communications, and rarely concern the automatic organization of the meetings. Then the diffusion of the multi-agent systems and the idea of personal assistant have given rise to theoretical analysis [5, 14] and to system-oriented approaches [9]: generally an agent-based approach has been adopted in which a personal agent performs some of the tasks involved in organizing meetings. The main focus of these approaches is on the automation of distributed scheduling and on particular negotiation aspects. More specifically, S.Sen's work is a comprehensive investigation on the heuristics for distributed problem solving, and on negotiation protocols for the meeting scheduling problem, aspects of which may be introduced in new versions of MASMA. Other systems have introduced the idea of agents as personal assistant [10, 4] and emphasized the automated learning aspects. Our approach is focused mainly on the agents' attitude towards the users: while some of the "structural" aspects of our solution can be also found in other systems, the attention on the user-agent relationship is particular of our system. The other systems usually do not account for properties as security and privacy, with the notable exception of [5], and rarely focussed on the specific problems generated by the interaction with the user.

3 MASMA's Architecture

MASMA proposes a solution in which the competence are distributed among different types of agents: moreover, inherently distributed problems are faced in a distributed way and centralized problems are faced in a centralized way. Its architecture consists of a personal assistant for each user, called *Meeting Agent*, and other three agents that are shared among a community. Each agent is an instance of the general model described in the next section. The MASMA's agents are the following:

- *Meeting Agent*. This agent is associated to each user and behaves as a personal assistant specialized in meeting organization. It has two main tasks: managing the user's profile and taking part in the meeting organization. The profile contains information concerning the personal agenda and the preference values assigned to the different dates and times: moreover it also maintains data about the user's general interests. In the organizational process the agent represents the user according to his profile.

The other three agents work as specialized knowledge servers to which some common services have been delegated. They could also perform their job autonomously without any connection with the Meeting Agents.

- *Server Agent.* It is in charge of managing the network addresses. In fact in an "open world" it is quite difficult that everyone knows all the addresses of everybody. A better solution is a specialized management in which a single agent maintains a knowledge base with the users' addresses: in the case of new users, it is able to get the addresses by querying one of the numerous servers. In MASMA it also manages a databases containing the interest areas associated with the users so that it can help the agent organizer to spread announces out in a selective way without bothering all the users connected. Moreover it also gathers the subscriptions in case of open meetings.

- *Resource Agent.* The Congress Centers or Universities or other similar sites are crucial resources in a meeting organization. MASMA adopts a centralized administration of these common resources to avoid conflicts in selecting one of them. Each site is characterized by $< attribute - value >$ pairs that describe it. The *Resource Agent* maintains the databases and furnishes to the Meeting Agent a list of structures satisfying the problem constraints. When a decision is taken, the agent carries out the operations necessary to reserve the place.

- *Travel Agent.* The user can also wish to mechanize the last step in organizing a meeting, the lodging and travel decisions. The *Travel Agent* can help the user is choosing the best path or the less expansive ticket or the most luxurious hotel. The agent can connect the user to train and flight timetable, decide the best path between to places, inform him about prices, show a list of possible hotels. It could also furnish a reservation service. At present it work on local databases, but it could also be connected to external information servers specializing the agent to understand database management languages.

MASMA is based on a mixed-initiative approach in which each step in any decision process is carefully considered: if the user prefers then any crucial decision remains up to him. Moreover he can always maintain the control on his agent and interfere and influence its actions. The user can dynamically influence the negotiation process by changing the constraints on line. After a testing phase, the user can decide to leave more and more decision steps to his agent although the possibility remains of inspecting and interfering in its behavior. The next version of MASMA will be endowed with learning capabilities: the agent will be able to discover rules from the user's behavior and to synthesize them after user's confirmation.

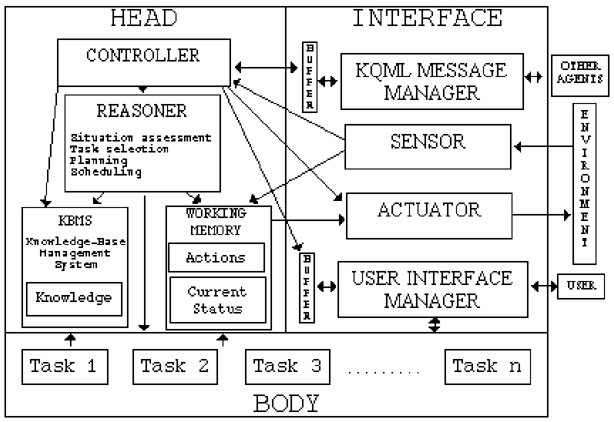


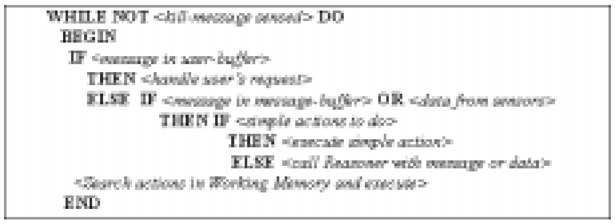
Figure 1: The architecture of the agents

4 A Model for an Agent Architecture

The proposed general model to design agents is flexible and adaptable enough to guarantee an incremental and modular development of the whole framework.
The architecture follows the `Body/Head/Mouth` metaphor [15], includes some ideas shown in other agent-based systems [6, 11, 8] modified by our experience in developing a previous system [3] (see Figure 1). It consists of three components:

- the *body* is in charge of task execution;
- the *head* accomplishes the reasoning and coordination tasks;
- the *interface* is devoted to the communication with the environment, the user and other agents.

The *body* carries out the specific tasks of the agent in the application domain. A task can consist of pre-existent software, public-domains program, on-purpose software or hardware components. That guarantees a high adaptability since it allows for incorporating and/or using any old part without developing from scratch. This part can autonomously access the data structures of the *head* and the communication facilities of the *interface*.
The *head* is devoted to coordinate the different functionalities; to manage the representation of the external world, of the agent and of the current state of affairs;



```
WHILE NOT <kill-message sensed> DO
  BEGIN
   IF <message in user-buffer>
      THEN <handle user's request>
      ELSE  IF <message in message-buffer> OR <data from sensors>
                THEN IF <simple actions to do>
                         THEN <execute simple action>
                          ELSE <call Reasoner with message or data>
   <Search actions in Working Memory and execute>
  END
```

Figure 2: The controller's cycle

to reason, to solve problems and take decisions; to coordinate and negotiate with other agents. In turn, it consists of four components, the controller, the reasoner, the knowledge base and the working memory.

- The *controller* coordinates the actions' flow. At present, it is a continuously-active process (Figure 2)—and so it satisfies the feature of temporal continuity for the agent—checking for messages from the environments or the user or other agents. If the current situation does not need reasoning processes, the *controller* is able to directly activate the requested tasks: otherwise the *reasoner* takes the *controller* over by putting the results of the reasoning process in the *working memory.* Then the *controller* searches the *working memory* for actions to be executed.

- The *reasoner* receives the messages coming from the external world that the *controller* cannot understand. According to the current state of affairs, described in the *current status*, to its knowledge and to its goals, the *reasoner* decides the actions or the plan to accomplish and their temporal sequence: then it puts them into the *working memory.* Moreover, it is in charge of updating and maintaining the *current status* and the *knowledge base* consistent.

- The *knowledge base* is managed by a deductive information retriever where information is represented as assertions and rules. It contains the user's mental attitudes, plan libraries, beliefs on other agents, on the external environment and on the application domain.

- The *working memory* is managed by a deductive information retriever. Since the agent can concurrently handle more situations, it is possible to instantiate different working memories.

The *interface* is in charge of the communication with the user, other agents and the environment. It consists of:

- The *KQML Message Manager* deals with the interactions between agents. It supplies the KQML [7] protocol and language, maintains buffers for input and output. Moreover it offers to the controller a set of functions for errors handling and for extracting the information content of the messages by excluding the communication parameters.

- The *Sensors* and *Actuators* allow the agent to exchange data with the environment. They typically consist of the instruction set of the operating system (Solaris 2.4). The actuators can influence the status of the environment: they can modify databases and/or activate printers, faxes, phones or other resources.

- The *User Interface Manager* communicates with the user following the user's preferred modalities. Since the system is addressed to any type of user including people not skilled in using computers, in the current implementation, the *User Interface Manager* supports a graphic language. It also manages a buffer for user's messages and transfers requests and data to the controller.

The subdivision of the agent's competence allows for separately facing with different types of problems emerging in the design of a specialized agent by supplying a *modular approach* to the implementation. It is possible to easily scale up all the components since the architecture allows for:

- incorporating pre-existent software or tools;

- defining the knowledge representation languages and/or the types of the mental attitudes;

- inserting further reasoning processes and defining the type of the control flow.

Consequently it is possible to incrementally develop multi-agent systems and to adapt them to different application areas. According to the application domain, it is possible to opportunely define:

- the dictionary of the messages exchanged and interpreted by the agents with their protocols;

- the specific functionalities for each agent;
- the sensors and the actuators;
- the interfaces for interacting with the user.

Moreover the potentiality of modifying the control flow gives a high level of flexibility to the architecture: it is possible to change the sequence and the synchronism of the actions according to the desired functionalities.

5 The Meeting Agent

A *Meeting Agent* is associated with each user given the personal nature of the handled data and the type of accomplished functions. At present, this agent is able to organize two different types of meetings, open meetings and closed meetings. In the first case, the organizer agent sends a sort of call for participation to the Server Agent that selects the possible participants on the ground of their interests. In the case of closed meetings, the agent sends the requests only to a set of people. In both cases a negotiation process can follows to decide the date and the place of the event. Nevertheless the agent is easily extendible to different types of conferences and appointments: it is already connected with the user's diary from which can extract and to which can communicate information.
The Meeting Agent can play the role of organizer or attendee by applying the correspondent negotiation protocol: it is possible to include more than one protocol and to base the choice on the context and current status. It is also possible to envision very sophisticated protocols that take into account several issues, for instance the role of the users in the community, the weight of previous engagements, the type of event, etc.
In order to mechanize the decision process and to limitate the interaction with the user, the agent maintains a user's profile with his availability and preferences and supplies tools to define and update the profile itself.

5.1 The User's Profile

The user's profile specifies the level of availability concerning the different time intervals: these values can be manually set by the user or deduced by the agent from the preference rules the user can define. The rules are described by a formal

language that is hidden to the user by a graphical interface in order to facilitate the interaction.
The user can assign a preference value {*high, medium, low, nil*} to each hour interval: the agent sets automatically to *nil* the dates in which appointments have already be fixed. The agent can take over the calendar system in the environment, in our case the calendar manager of the UNIX systems. For the hour intervals the user have not any appointments, the agent figures out the value according to the preference rules. There are three different types of rules it can apply:

1. The `Holiday-Rule` associates a preference value to any weekend day and holiday. So if the user likes to work on Sunday morning, he can assign the value "high" to this interval. The holiday are automatically extract by the calendar, but the user can also set his own holiday time.

2. The `Proximity-Rule` allows the user to give a preference value to a time interval around a previous meeting. For example, it is possible to specify the time distance between two different appointments in the same town, e.g., 2 hours, or in different towns, e.g., two days.

3. The `Fixday-Rules` can be used to define an availability value for the same day(s) of the week or for the same day(s) of a month in a specified time interval: for example, the user can be busy every Monday at 9 until 11 from June 1996 to July 1997 or every 15 in the same time interval.

Starting from the user's preferences, the calendar setting and the rules, the agent calculates the final value by applying an algorithm that also accounts for inconsistencies introduced by the user himself.

For sake of simplicity, the user communicates to his Meeting Agent his personal interests concerning both his work and his spare time. Then this agent sends to the Server Agent this information so that the user can be informed about the events regarding his interest areas. For example in the case of open meetings and call for papers, the Server Agent sends the announcement only to the users having among their interests the topic of the conference.

5.2 A Basic Negotiation Protocol

In some cases it is straightforward to get a date for an event, but more often it is quite complex to find a middle course: so a negotiation process is engaged by the organizer agent in order to reach an agreement. The attendants' agents try to protect

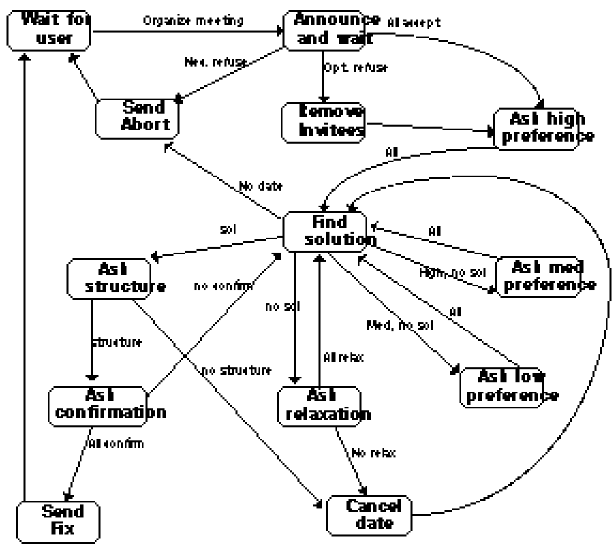


Figure 3: The organizer's negotiation protocol

their users and they in turn apply a strategy to safeguard their privacy and to avoid the relaxation of important constraints. At present, the strategies are fixed—one for the organizer and one for the participants—but it is possible to differentiate them. The agent organizer has the goal to look for an optimal solution: obviously the optimality depends on the selected criterion. At present the solution is to maximize a common utility function and to minimize the requests for constraints relaxation.
Once the user has defined the features of an event (according to the problem description shown above), his Meeting Agent sends the announce to the interested users through the Server Agent. At this point the negotiation begins, and the organizer agent behaves according to a specified protocol (Figure 3).
The agents receiving the announce calculates the availability of their users concerning the proposed date(s) and then can ask for confirmation to them: this step can also be mechanized. It is possible that some of the invitees rejects the request: if he is an optional guest then the negotiation goes on otherwise the process interrupts. The organizer can decide to re-arrange the event by changing its features. This is still a limitation of the agent that does not include a full recovering mechanism. In the case that all the necessary participants accept, the organizer agent asks the other Meeting Agents for the time interval with an high preference values: if it does

not find any possible intersection, it asks for lower preference values. If it is still not able to find any possible data, it starts asking other agents for relaxing some dates. In order to fasten the converging of the decision process, the agent suggests the interval with the highest number of adhesions and with the highest total preference values: it asks to relax it only to the agents having previously rejected that date. If they accept, then they supply a new preference value for that time. Whether at least one of the them rejects the proposal, the step is applied again with the successive intervals until an agreement is reached. It is also possible that some of the participants decide not to participate any more, or that the organizer asks for a decision to the agents obstructing the negotiation.
It is be noted that the process can cause side effects: a Meeting Agent can be constrained to cancel a previous engagement and so it has to start a new dialogue with the other agents involved in order to re-schedule it. Before taking a so critical decision, the agent asks for confirmation to its user. Also this step can be mechanized: for example, the user could have instructed his agent (or it can have learned such a rule) that the request of his boss have absolute priority with respect to any other appointment including personal engagements. These changes will appear in the next version of MASMA.
When the organizer agent has a date, it inquires the Resource Agent to verify the availability of the sites: it sends it the part of the problem description concerning the $< Place >$ and $< Instrument >$. It could be necessary to change some features if the chosen place is not available. It could also be necessart to look for a new date. When the place has been booked, the Meeting Agent confirm the event to all the participants. They can interrogate the Travel Agent to decide how to get the place and the available hotels: the agent may reserve for them.
At any step of the negotiation process, the users can modify their availability—either manually or with the rules—and so influence the decisions. Moreover, a user can decide to re-negotiate the meeting even after the final setting: probably the other users could not agree with him!

6 Interacting with the User

For systems that work as personal assistants, and more generally for those devoted to help managing tasks, the attention paid to the interaction with the user makes the difference with respect to their actual use. Although we do not consider our system a final answer to the problem, we claim that some of the aspects we have

considered are a step in a relevant direction and distinguish us from other distributed approaches to the meeting scheduling more concerned about the specific distributed scheduling problem.
The agent-based approach in designing useful and effective support tools cannot be separated from developing active and cooperative interfaces able to mediate between the user and the application. From the designer's standpoint, the agent is the interface and so it is supposed to fasten the user's delegation of decisional steps to it. The interface should acquire a proposer role with respect to the utilizer's needs and preferences: in this context the part of *task delegation* plays a relevant role in which the trust relationship agent-user becomes subject of research. In fact a drawback of agent-based systems is in the user's acceptance and in the consequent user's mistrust in delegating his choices. The agent should substitute the user in the routine parts but leave him the more crucial decisions. Norman [12] enlisted several issues to be taken into account in agent-based systems:

- ensuring that the user feels in control of the system;
- considering the nature of agent-user interaction, how the user teaches his agent, how the agent and its activities are presented to the user;
- implementing safeguards to cut off or minimize the effects of agents' errors;
- giving the user accurate expectations respect to the system;
- accounting for the user's privacy particularly in applications in which the agent accesses personal data and mail;
- hiding computational complexity but at the same time making the system inspectable.

Many of these issues are few or not at all considered in current literature despite the increasing number of system proposed as *interface agents.*
In MASMA particular attention has been paid to issues that are essential to the user's acceptance of the system like: the possibility of inspecting the agents, the respect of the data privacy, the simplicity of utilization of the whole tool.
A major drawback that the HCI community ascribes to agent technology is the user's loss of control about the activities of Software Agents. Indeed some reasons for this complain exist because of the limited research dedicated to examining the role of a Software Agent tool as a mixed-initiative interactive system opposite to a

completely automated system. Although exciting, the possible autonomy of software agents should be held under control in systems that continuously interact with the user, in spite of the requested autonomy for agent-based active interface. An agent should be endowed with the capability of acting and autonomously proposing solutions according to the current problem, but the user must have the possibility of controlling and inspecting the agent's decisions. To this purpose we have investigated on the *possibility of inspection* of the agent behavior from the user. The agent-based system should be endowed with an inspection mode about its activities, take the user into account in the most important decision processes and be autonomous in the less critic decision steps: that will involve a major level of trust by the user. MASMA allows in its main dialogue window to verify the status of the agent, and then to inspect the details of the current activity. When no negotiation is running the agent is free, and it is busy when it is involved in a organizational process. In this case, a button is active that allows the user to analyze the running processes, to verify the information and data at agent's disposal and to interfere or take over the agent if necessary: moreover the user can influence the organization and negotiation processes by dynamically modifying the preference and availability values. This capability of the system is a first step in the development of mixed-initiative system.
From the point of view of user involvement, attention should be paid to the decision about *how* and *when* to interact with the user. Concerning *how* to interact, the system must be *easy to use*. The agents dialogue with their users by means of simple and effective windows that can be adapted to the current needs and that hide the complexity of the system. To be able to relieve the user's work, the interfaces should be few invasive and really discreet. MASMA's interfaces satisfy these requirements. The second point concerns *when* the agent calls for its user. The agent does not have to be invasive but at the same time it is active and takes decisions that could be critical. The personal agent is supposed to always involve the user in decisional stages but a system that continuously asks for confirmation would fail in its main task. That can happen independently from the possibility of taking over the agent. The meeting agent interrupts its user only for the most important—in user's view—decisions: this means that is the user who decides when to leave decisions to the agent.
Another relevant feature is to respect the *preservation of information privacy*. In the described scenario the personal agent can access strictly confidential data (mail, diary, personal data, etc.) so it is particularly important that such data are manipulated carefully and protected from intrusion. In order to increase the user's

confidence level in the system, under the user's control MASMA always follows a policy that makes available only the dates strictly necessary with the highest preference value. Requests for new dates or for relaxing time intervals are made only when there is no other solution. Moreover, to limitate the number of people requested to relax constraints, the agent always chooses the interval the highest number of participants agree on and with the highest preference value.

7 Conclusions

In this paper we presented MASMA, a multi-agent system devoted to interact with a set of users to help organizing meetings. To avoid producing "yet another agent-based meeting scheduler", we have focussed our attention on the interaction with the user in order to obtain an effective mixed-initiative system.
In the paper we have described both the general aspects of the system and the issues concerning the agent-user relationship.
The system is being tested on real environments and has been applied to a reference problem like the above mentioned *secretaries' nightmare* (see [1]) with satisfactory results.
Our approach can either represent a comprehensive solution to the whole problem or be seen as a solution containing aspects that can be integrated with existent approaches.

References

[1] R. Brancaleoni, A. Cesta, and D. D'Aloisi. MASMA: un sistema multi-agente per l'organizzazione automatica di appuntamenti (in Italian). In *Proceedings of the Italian Annual Conference on Computer Science (AICA 96)*, Rome, Italy, September 1996.

[2] A. Cesta and D. D'Aloisi. Building Interfaces as Personal Agents: a Case Study. *Sigchi Bulletin*, 1996. to appear.

[3] A. Cesta, D. D'Aloisi, and V. Giannini. Active Interfaces for Software Tools. In Y.Anzai, K.Ogawa, and H.Mori, editors, *Symbiosis of Human and Artifact (Proceedings of the 6th International Conference on Human-Computer Interaction)*, pages 225–230. Elsevier Science B.V., 1995.

[4] L. Dent, J. Boticario, J. Mc Dermott, T. Mitchell, and D. Zabowski. A Personal Learning Apprentice. In *Proceedings of the National Conference on Artificial Intelligence*. MIT Press, 1992.

[5] E. Ephrati, G. Zlotkin, and J.S. Rosenschein. Meet Your Destinity: A Non-manipulable Meeting Scheduler. In *Proceedings of the CSCW Conference*, pages 359–371, 1994.

[6] O. Etzioni and D. Weld. A Softboat-Based Interface to Internet. *Communication of the ACM*, 37(7):72–76, 1994.

[7] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, R. Pelavin, S. Shapiro, and C. Beck. Specification of the KQML Agent-Communication Language. Technical report, `http://www.cs.umbc.edu/kqml/`, February 1993.

[8] N.R. Jennings. Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. *Artificial Intelligence*, 74(2), 1995.

[9] A. Lux. A Multi-Agent Approach towards Group Scheduling. Technical Report RR-92-41, DFKI, 1992.

[10] P. Maes. Agent that Reduce Work and Information Overload. *Communication of the ACM*, 37(7):30–40, 1994.

[11] J.P. Müller and M. Pischel. An Architecture for Dinamically Interacting Agents. *International Journal of Intelligent and Cooperative Information Systems*, 3(1):25–45, 1994.

[12] D.A. Norman. How Might People Interact with Agents. *Communication of the ACM*, 37(7):68–71, 1994.

[13] C. Petrie, editor. *Proceedings of the CAIA Workshop on Coordinated Design and Planning*. `ftp://cdr.stanford.edu/pub/caia-wrkshp/`, 1994.

[14] S. Sen and E.H. Durfee. A Contracting Model for Flexible Distributed Scheduling. *Annals od Operational Research*, 1996. to appear.

[15] D.D. Steiner, D.E. Mahling, and H. Haugeneder. Human-Computer Cooperative Work. In *Proceedings of the 10th International Conference on Distributed Artificial Intelligence*. MCC Technical Report ACT-AI-355-90, 1990.