Agent-based Meeting Scheduling: A Design and Implementation

N. R. Jennings and A. J. Jackson
Dept. of Electronic Engineering,
Queen Mary & Westfield College,
Mile End Rd.,
London E1 4NS.

Indexing terms: multi-agent systems, distributed meeting scheduling.

This letter describes the design and implementation of a distributed meeting scheduling system in which each user has an intelligent agent in their computer desktop which is responsible for arranging meetings. Knowing the preferences and commitments of their user, the agents negotiate with one another to find the most acceptable meeting times.

Introduction: It is widely believed that the next generation of computer desktop applications will be significantly more proactive in helping users achieve their goals than those which currently exist [1, 2]. Rather than the user having to specify each and every step of a given task, the desktop of the future will be composed of a series of *intelligent agents* to which a number of high level tasks can be delegated. These agents will be responsible for autonomously deciding how the task is to be achieved and actually performing the necessary set of actions (including handling possible interactions with other intelligent agents). Examples of the types of functionality which will be supported include: filtering electronic mail, monitoring Internet newsgroups and reporting back interesting discussions, and discovering new data repositories on the information superhighway. This letter reports on the design and implementation of a particular agent-based application which arranges meetings. This application turns the currently available electronic calendar management systems (such as Organiser or UNIX's Calendar Manager) from passive information repositories in which a user simply records his personal schedule into active and empowered applications which can negotiate on behalf of their user to arrange meetings according to a set of expressed preferences. This work represents an advance over previous research prototypes (eg [3, 4, 5]) in that it deals with the postponement and rescheduling of meetings as well as their effective scheduling in a dynamic and changing environment. It is an advance over currently available calendar management products (such as Meeting Maker and MS-Schedule+) in that it is the agents, and not the humans, which manage and enact the negotiation process.

Agent Design & Implementation: A centralised database which stores the calendars of all the participants in the system is infeasible because it violates the privacy of the individuals concerned. For this reason, it was decided that individuals should maintain their own calendar locally and hence they should each have a meeting scheduling agent (MSA) acting on their behalf (fig. 1). The system is invoked when a host indicates to its MSA that it would like to schedule a meeting involving a number of individuals (eg B and C) at a specific time (eg "one afternoon this week"). The host's MSA announces the meeting, together with the associated constraints, to the proposed invitees. However since the host's MSA does not know the commitments of the invitees, the proposal may conflict with the invitees' existing arrangements. In such cases, the MSAs negotiate with one another to resolve the conflict - possible means include cancelling the meeting, rescheduling the meeting or rearranging existing meetings. At all stages in this process, the MSAs take their user's preferences into account (eg no meetings at the weekends and all meetings between 9 and 5).

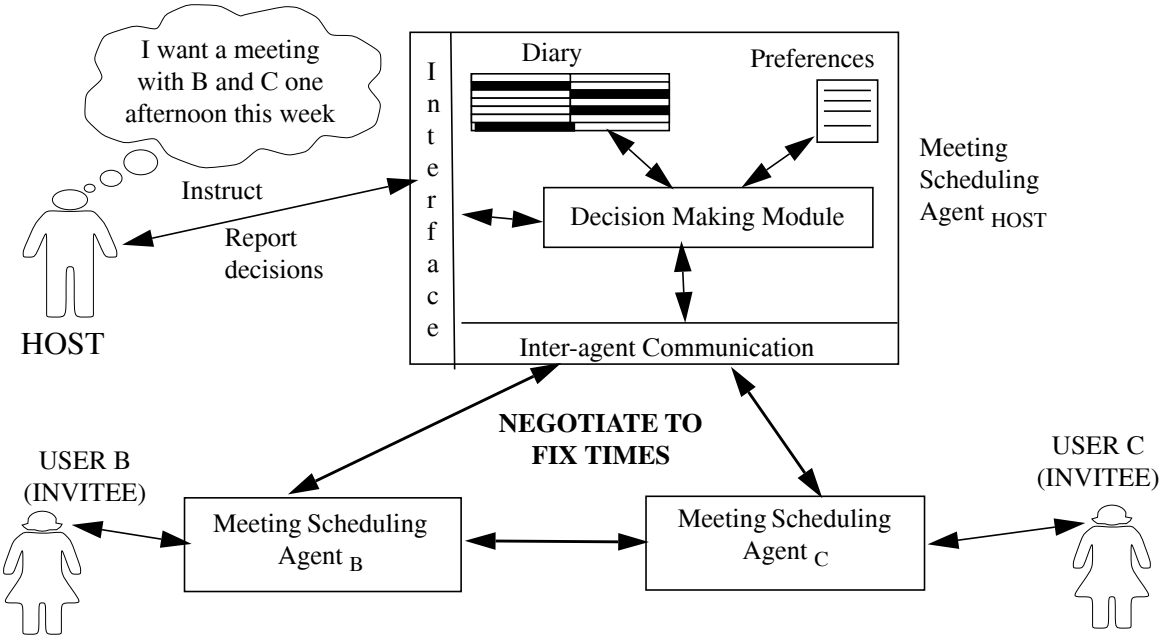

Figure 1: Agent-based meeting scheduling: System Design

The algorithm encoded in each agent's decision making module is given in fig. 2. It is a two phase algorithm - in the first phase (steps 1 & 2) the invitees provide information about whether and when they are willing to attend a particular meeting and in the second phase (steps 3-5) an actual meeting time is proposed and (hopefully) agreed upon. There is potential for several rounds of iteration in each phase if the specified objective is not satisfied immediately.

1: Host (H) for meeting M_j sends out invitations to all invitees $\{I_1....I_n\}$ indicating the purpose of the meeting, the constraints which apply to the meeting (C_{Mj}), and the desired length of the meeting (L_{Mj})

2: FORALL $I_i \in \{I_1...I_n\}$ DO: Determine potential times $\{T_{Mj,Ii,1}....T_{Mj,Ii,k}\}$ at which Mj could be scheduled (according to C_{Mj}, L_{Mj}, and local preferences) and how many of these options (m) will be offered to H. Make tentative commitments to these times $\{TC_{Mj,Ii,1}..... TC_{Mj,Ii,m}\}$. Return bids to H.

3: IF there exists a time slot $<T_{Mj,Ii,q}>$ present in bids from all the invitees and its global preference is above the host's threshold for M_j
THEN select slot (Ψ_{Mj}) with highest global preference
seek confirmation from all attendees for M_j at Ψ_{Mj}
ELSE IF further times available for M_j
THEN ask for further bids for M_j and repeat from stage 2 ELSE replan M_j

4: FORALL $I_i \in \{I_1....I_n\}$ DO: Respond (confirm or reject) to proposal from H for M_j at Ψ_{Mj}

5: IF H receives sufficient positive responses to proposal Ψ_{Mj} to make M_j viable
THEN schedule M_j at Ψ_{Mj} ELSE replan M_j

Figure 2: Distributed Meeting Scheduling Algorithm

In more detail, the request for a meeting is initiated by the host agent (Agt_A) who specifies the meeting's attributes (see below) - these include a list of the desired attendees together with the associated constraints (eg all invitees must attend), the objective and length of the meeting (in hours), and the time window during which the meeting must take place. This time window may vary from the very specific (eg Friday 15:00) to the very vague (eg "some time Friday afternoon").

(FROM Agt_A TO (Agt_B Agt_C) (MEETING-INVITE (OBJECTIVE BRAIN-STORM)
(LENGTH 3) (CONSTRAINTS (FRIDAY PM) (ALL INVITEES ATTEND))))

Upon receipt of this invitation, each invitee decides whether it is interested in attending the meeting. Assuming an agent is willing, it sends a ranked list of m possible times together with their respective preference ratings back to the host (see below, where m=2 and midday is preferred to late afternoon). The agent also makes tentative commitments in its diary for these times. A tentative commitment indicates that the time slot may be used for that meeting (hence it should be less favoured than an empty slot when arranging subsequent meetings) but it can be overwritten by tentative commitments to a higher priority meeting or any firm commitment.

(FROM Agt_B TO Agt_A (POTENTIAL-MEETING-SLOTS BRAIN-STORM
(FRIDAY 12:00 RATING 4) (FRIDAY 16:00 RATING 2)))

In the current implementation m is fixed for a given agent, although in the general case the agent should be able to dynamically determine the number of responses it makes depending on its rating of the importance of the meeting, its relationship to the host, and the meeting scheduling requirements (the more bids it returns the more likely there will be an agreement). In order to determine which slots to offer to the host, each invitee computes an ordered list (as a function of preference) of all its available meeting slots which satisfy the constraints and timings expressed in the meeting proposal. An agent's preference for a meeting is the sum of the preferences for the individual time slots involved in the meeting divided by the duration of the meeting (if the time slots are uncommitted). For any slots which are either tentatively or firmly committed, the agent's preference for that slot is reduced in proportion to the priority and type of the existing commitment (firm commitments are reduced more than tentative ones).

Once the host has received all the potential meeting times from the invitees, it tries to find a mutually agreeable slot. In the current implementation this is done by searching for an intersection among all the offered bids. If such an intersection exists, its global preference is computed, by averaging the offered preferences, and if this value is higher than a preset host-defined threshold the slot is deemed acceptable (if there is more than one acceptable slot, the one with the highest global preference is selected). However if no acceptable slot exists there is a further iteration of this phase of the algorithm. The host stores the offered bids in its local database and asks the invitees who have not yet offered all their available time slots whether they would like to add to their original m proposals. If new proposals are forthcoming they are evaluated as before to see if a valid intersection now exists. This continues until all invitees have offered all their bids at which point the host realises that no mutually acceptable time exists and so the meeting should be replanned. Replanning involves altering the meeting's parameters - either by changing the meeting's duration or time window, by inviting different attendees or by changing the constraints on attendance.

Assuming the host is able to find an acceptable time, it sends out a confirmation message to each of the invitees (see below). Each invitee then has to either confirm or reject the meeting proposal. When making this decision the agent is faced with four possible situations: (i) its tentative commitment for M_j at Ψ_{Mj} can be made firm because there are no conflicting obligations; (ii) the slot at Ψ_{Mj} has been firmly

committed to a more important meeting M_k and so the proposal for M_j is rejected; (iii) the slot at Ψ_{Mj} has been tentatively committed to a meeting M_k in which case the proposal for M_j is accepted; (iv) the slot at Ψ_{Mj} has been firmly committed to a less important meeting M_k in which case the proposal for M_j is accepted (as for iii), but in addition to this a request is sent out to M_k's host to reschedule that meeting. This request may be granted or denied by M_k's host - it is granted only if the complexity of replanning M_k is sufficiently low (currently defined by the number of iterations required to reach agreement), otherwise M_k has to be replanned (as described above).

(FROM Agt_A TO (Agt_B Agt_C) (MEETING-PROPOSAL BRAIN-STORM
(FRIDAY 16:00) (LENGTH 3))

When the host receives all the responses to its proposal it must determine whether the meeting is viable. This decision depends upon the constraints on the attendees. In the example shown, all attendees must be available for the meeting to be viable, however in the general case the host may consider the meeting viable if a certain percentage of the attendees can come or if named important individuals are able to attend.

Conclusions and Future Work: This letter described the design and implementation of an important agent-based application, namely a distributed meeting scheduling system. It outlined a novel scheduling algorithm which deals with the dynamics inherent in a busy office environment and also allows meetings to be postponed and rescheduled. For the future, a systematic empirical evaluation of the options expressed in the scheduling algorithm is planned so that the benefit of the differing commitment policies can be assessed.

References

1 E. Germain (1994) "Software's special agents" New Scientist, 142 (April), pp 19-20.

2 N. R. Jennings (1994) "Cooperation in Industrial Multi-Agent Systems" World Scientific Press.

3 F. Mattern and P. Sturm (1989) "An Automatic Distributed Calendar and Appointment System" Proc. of EUROMICRO Conference, Amsterdam, The Netherlands.

4 S. Sen and E. H. Durfee (1992) "Automated Meeting Scheduling among Heterogeneous Agents" Proc. AAAI92 Workshop on Cooperation Among Heterogeneous Agents, San Jose, CA., pp 116-120.

5 N. Eisinger and N. Elshiewy (1992) "MADMAN - Multi -Agent Diary Manager", Proc. of DAI Workshop at European Conference on Artificial Intelligence, Vienna, Austria.