

Development of a personal agenda and a distributed meeting scheduler based on JADE agents

Miguel Ángel Sánchez
Álvaro Rayón Alonso

Grupo de Sistemas Inteligentes
Departamento de Ingeniería Telemática
Universidad Politécnica de Madrid

Abstract: In this article we are going to describe the development of a multi-agent system based on the JADE platform. This system is integrated in the COLLABORATOR European project which aims to provide a set of services to facilitate collaborative work among users. These users, the applications and the documents they share define the key concept of collaborative session. The multi-agent system helps the users to create and manage these sessions. The services provided by this multi-agent system are a distributed meeting scheduler and a personal agenda manager.

1. Introduction

The COLLABORATOR (COLLABORative fRAMework for remoTe and mOBile useRs) IST-2000-30045 [11] project is designed to provide a shared workspace supporting mobile users' activities accessed through different types of devices and heterogeneous communication networks. COLLABORATOR is delivered as a web application which will run inside a standard J2EE application server. Collaborative applications are platform independent Java classes that can be loaded dynamically and run by a web server supporting particular services. COLLABORATOR features are:

- Multiple collaborative parallel sessions.
- Application Sharing: this service provides
 - The sharing of off-the-shelf Java and Windows applications allowing users to act together on the same instance of the application.
 - A specialized software module to share Microsoft PowerPoint HTML presentations.
- Personal Agenda Management: this application allows the users to enter/modify/cancel appointments, to visualize this information in a friendly way and to synchronize with other agenda clients.
- Meeting Scheduler: this service allows a coordinator to negotiate a session date with the invitees taking into account the time constraints introduced by the coordinator and the time usage preferences of all the attendees. It also manages the life-cycle of a collaborative session.
- Chat: interchange of short text message in real-time among the users of a session.
- Multimedia services: these services provide videoconference.
- Personalization of collaborative session views.

As we have said before the personalization and adaptation of the platform to the user preferences is an important target. To cope with the requirements related to the management of the collaborative sessions a distributed multi-agent approach has been chosen [4] [5] [6]. The agents focus on helping the user to create sessions by negotiating the best suitable date for this appointment, taking into account each user's scheduling preferences and the actual status of his personal agenda. The optimal solution to this problem is the one that maximizes the number of attendees and satisfies the time constraints imposed by them [3] [8] [9]. Once the collaborative session has been fixed, the multi-agent system will be in charge of process all the events related to these sessions as who enters and leaves a session and when they do it, session cancellation, attendees' assistance changes, notifications about the session starting/ending event. These last three events fire actions like synchronize the attendees' personal agenda or send an email indicating the changes affecting a session.

2. Agent Subsystem

2.1 Functionality

The agent subsystem is the COLLABORATOR framework component that manages the personal agenda and meeting scheduling services. In order to achieve this, two different type of agents have been developed.

Personal Agent (PA)

Each system user has a Personal Agent in charge of manage his personal agenda. This agent obtains the personal profile monitoring the user interaction with this agenda. The profile will contain information about:

- a) Meeting categorization: each collaborative meeting is defined by the following properties –attendees (with his relative importance), textual subject and description, several time intervals that defines the time constrains, the duration and a set of keywords defining the expertise required on that meeting-. Based on this information, a user can define classes to classify collaborative or personal meetings. Observing the user's actions, the PA would be able to autonomously categorize meeting request. This categorization will allow the agent to infer when the user prefers a meeting to take place and communicate these preferences to the date negotiation process.
- b) Time preferences: the COLLABORATOR users can instruct the agent about how their personal agendas should be managed by giving temporal constrains and preferences expressed by rules like “I don't want any meeting on Mondays and Wednesdays mornings (or between 10 and 12 AM)” or “I prefer Tuesdays and Thursdays evenings for meetings of class SALES_MEETINGS”. The PA will also infer using and adaptive learning algorithm these rules based on the personal calendar state and user's scheduling actions.

With all this information the PA will be able to schedule autonomously the personal appointments of his user and help him in planning the collaborative ones. The users can activate/deactivate his personal agents at will. This indicates his disposition to receive collaborative meetings requests from other users.

Session Management Agent (SMA)

Each time a new collaborative session is created by a session coordinator, a new Session Agent is associated with it. When this happens, the SMA starts a negotiation process with all the session invitees through their PAs trying to determine the best suitable date for the session. Based on the meeting time constraints imposed by the coordinator, all the active PAs inform the SMA about at which hour their users prefer the meeting to take place or if they don't want to attend to the meeting. When the SMA finishes gathering all the responses, it decides if the meeting can take place and the best date for it. The coordinator is informed about this decision so he can evaluate it. If the coordinator confirms the final date, the SMA informs all the PAs about the event so they can synchronize their users' personal agendas and inform them about the new meeting. Once the session is created the SMA will be in charge of managing all the subsequent changes related to that session as:

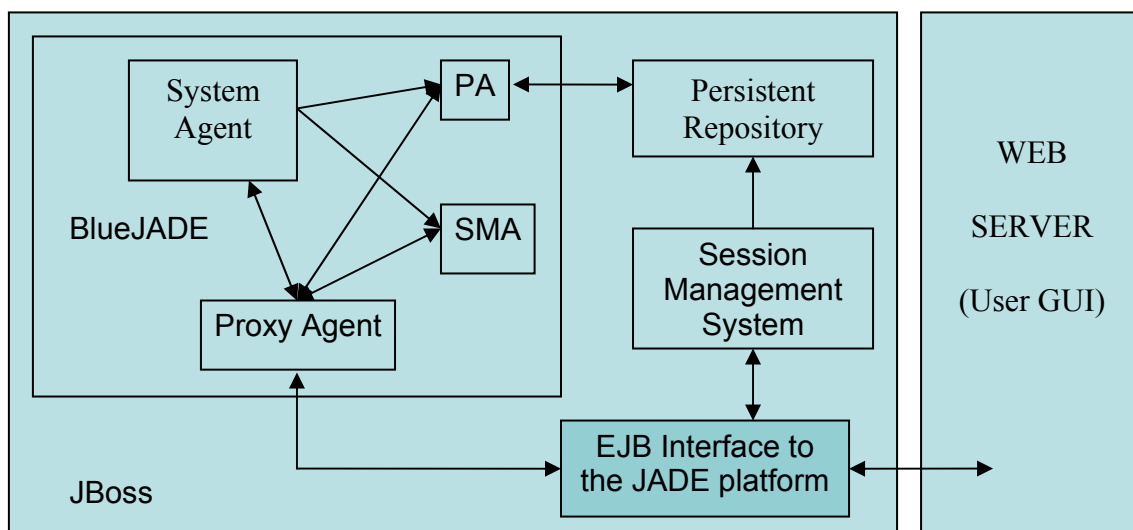
- An attendee decides to cancel/confirm his assistance directly.
- An attendee decides to propose a substitute for him. When this happens the substitute PA is invited to the session but no new date negotiation takes place.
- The session is cancelled by the coordinator.
- A user enters/leaves the session: this information will be provided to a log service to keep track of user's attendance.

System Agent (SA)

This agent performs general system related actions like searching periodically for sessions about to start, informing the related PAs and SMA and locating running agents that meet specific characteristics.

2.2 Architecture

The following figure represents the agent subsystem structure:



As we can see in the figure the JADE platform has been integrated in the JBoss application server (J2EE container) as a service. This has been accomplished by using BlueJADE [2] extension of JADE. We have come to this solution because the rest of the framework components were developed as EJB and this improves the average system performance as all of them run inside the same Java Virtual Machine. BlueJADE also

allows JBoss to manage the agent platform life-cycle. It starts the agents at boot time, start/stop the platform without stopping the application server and log the events generated by JADE (platform failures, agent creation and destruction, etc).

One drawback of the agent paradigm is that the asynchronous message-driven agent communication model doesn't fit well with the most conventional synchronous blocking method calls model used by applications developed with the client-server design pattern. For example, if a system component needs a service which is achieved by the interaction of a set of agents and the client needs to block waiting for the result, a synchronization mechanism between the client running outside the agent platform and the platform itself is needed. This problem is aggravated in our runtime environment because we don't have any control on the threads executing the services inside the server application. When a COLLABORATOR component needs a service provided by the agent platform it executes a method call on an EJB session bean with a well-defined interface. This EJB wraps all the communication processes as, creating a Fipa-ACL request message, send it to the right agent through the Proxy Agent, wait for the response and give the result back to the calling component. All the interactions with the agents must be done using ACL-messages. As the entities running outside the JADE platform are unknown to the agents, they can't send response message directly to these entities. This is why the Proxy Agent is needed. It provides the "illusion" of a blocking synchronous method call as follows:

- At boot time the Proxy Agent is started and it creates a server-side socket, binds to it and waits for requests.
- An EJB executes a method call on the Session Bean Interface to the agent subsystem.
- The Session Bean Interface builds a Fipa-ACL message with all the information needed to perform the request and with the agent destination address. It opens a socket (action allowed by the EJB standard) and sends the message to the Proxy Agent. Then it blocks waiting for the response for a fixed amount of time. If a timeout happens, the calling EJB is informed about this event (synchronization mechanism).
- The Proxy Agent receives the request message and creates a new behavior to serve it. It modifies the message by registering itself as the sender of the message and then it sends it to the destination agent.
- The destination agent performs the needed tasks and returns his response message to the Proxy Agent.
- The Proxy Agent gets the content of the response message and builds a new one, sending it back to the Session Bean Interfaces through the opened socket. It then closes this socket.
- The Session Bean Interface unwraps the content of the response message and returns the result.

2.3 Services

The agent subsystem, working on behalf of the user, enhances the functionality of several COLLABORATOR framework services. This subsystem tries to take decisions autonomously like the ones that the user would take in the same situation.

The services provided by the agents are mainly related with the customization of the Personal Agenda and the management of the collaborative session, with special emphasis in the Meeting Scheduling service.

Personal Agenda

This service offers the basic functionality of an agenda. It stores, deletes, classifies the personal and collaborative appointments of the users. It is also able to synchronize with other agenda applications via the VCalendar standard.

The PAs are capable of answering questions about when the user is available, when he prefers a specific type of collaborative session and scheduling a personal meeting based on a set of time constraints. To cope with these functionalities, these PAs try to infer, in a transparent way, a general model of their users' behavior while they are using the application [5][7][10]. Despite of this, the user has a complete control of the decisions taken by his PA by means of editing his personal profile using a friendly interface [1].

The personal profile is classified in two categories:

- Information explicitly given by the user.
- Information inferred by the PA

If the information inferred by the PA doesn't match with the information provided by the user, this last one will be chosen.

The personal user profile is obtained as follows:

- Meeting categorization: the PA tracks every meeting classified by the user. It builds four bag-of-words for each meeting class. These bag-of-words are vectors whose components are the frequency of the terms appearing in all the meeting of a specific class. There is one vector for each of the following attributes describing a meeting: attendees, subject joined with description, expertise required and location of the meeting. When a new request for a collaborative meeting arrives, with the terms appearing in the request the PA will build a second set of vectors. When building them we remove the stop-words and some predefined vocabulary and stem all the subject and description terms. Finally, using information retrieval techniques (tf-idf and cosine distance), these vectors are compared searching for the maximum class similarity. If the user confirms the result of the categorization, all the terms appearing in the meeting will be added to the bag-of-words of the resulting category. If not, the terms will be added to the category selected by the user.
- Time preferences: the information taken into account for the learning algorithm can be divided in three classes:
 - Information related to the appointment itself: moment of the day (morning, evening, afternoon, etc), day, week of the month, month, category (see previous paragraph), duration, etc.
 - Information related to the state of the personal agenda at the moment of the appointment setting: type of the meeting after/before the one we're analyzing, time available in that day, etc.
 - Information provided by the user related to the degree of mobility i.e. if a specific meeting can or can not be moved to another date if necessary.

In general, we are including any parameter that could be related to the user's scheduling decisions. From all these information we infer the time preferences with the learning algorithm described below [7]:

- Periodically the PA will build all the training data from the personal agenda taking into account all the information described above. This data will be taken from the most recent time interval (a window from a predefined number of days in the past until now).
- An induction learning algorithm (like ID3) will be used to find a set of rules from the training data.
- All these rules will be compared with the existing user profile rules to find out if there is already a most general one. The more specific rules will be dropped.
- The rules that will never be used (because they refer to situations in the past) will be dropped too.
- The new rules inferred will be added to the personal profile.

Each time a user sets a meeting or receives a collaborative one, the PA will test if the learned rules match the new user decision. Each rule has two counters: one for the number of times that it has been fired and another for the number of times it has guessed right. If the guessed-right/fired ratio falls below a threshold the rule is dropped from the personal profile. When there is a collision between several opposite rules the one with higher guessed-right/fired ratio will be chosen.

Meeting Scheduler

Each time a coordinator sets up a new collaborative meeting, a new SMA is created with all the information provided by him (including attendees, time constraints, etc.)

This SMA will ask all the participants' PAs to send every time slot satisfying the time constraints with each user's likelihood to attend the meeting (with a number between 0 and 1). With all these results, the SMA will try to find a date that maximizes the number of attendees. In the case of several possible dates, the SMA will select the one with the biggest mean. This mean is obtained from the numeric values provided by the PAs for each slot.

3. Conclusion and Future Plans

In this article we have proposed one possible solution to obtain personal profiles based on user interactions with a specific application (personal agenda). These profiles are then used to guide the behavior of the Personal Agents that work on behalf of their users. We have also wanted to stress the difficulties to integrate an agent system based on the JADE platform with applications that runs outside the JADE environment and don't use an asynchronous message-driven communication paradigm.

Future Plans

We are planning to incorporate to the agent subsystem the ability to classify users on the basis of its knowledge. This will allow finding users with deep knowledge in specific areas by querying the system. The agent subsystem will try to provide lists of users who satisfy the requirements that a COLLABORATOR user is looking for.

The personal profile has some key words that are selected by the user from a closed set. These key words describe the expertise of the user. The expert searching (classification of the users) will be based on the users' profile key words and in the collaborative meeting they attend to.

Acknowledgements

This work is partially supported by the European Commission through the contract IST-2000-30045, COLLABORATOR (COLLABORative fRAMework for remoTe and mObile useRs) [11]. The authors wish to thank all the participants in this project.

References

- [1] A.Cesta and D. D'Aloisi. Mixed-Initiative Issues in an Agent-Based Meeting Scheduler. *International Journal on User Modelling and User-Adapted Interaction*, 9(1/2):45-78, April 1999
- [2] Dick Cowan, Martin Griss. Making Software Agents Technology available to Enterprise Applications. Available at <http://www.hpl.hp.com/techreports/2002/HPL-2002-211.html>
- [3] Leonardo Garrido and Katia Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In *International Conference on Multi-Agent Systems*, pages 95-102, 1996
- [4] N.R. Jennings and A.J. Jackson. Agent based meeting scheduling: A design and implementation. *IEEE Electronics Letters*, 31(5):350-352,1995
- [5] Robin Kozierok and Pattie Maes. A learning interface agent for scheduling meetings. In *Intelligent User Interfaces 93*, apges 81-88, 1993
- [6] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31-40, 1994
- [7] Tom Mitchel, Rich Caruana, Dayne Freitag, John McDermott and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81-91, 1994
- [8] Sandin Sen. Developing an automated distributed meeting scheduler. *IEEE Expert*, 12(4):41-45, July/August 1997
- [9] Sandip Sen and Edmund Durfee. A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, 7:265-289, 1998
- [10] S. Schiaffino, A. Amandi. On the Design of a Software Secretary. *Proceeding of the Argentine Symposium on Artificial Intelligence*. Santa Fé, Septiembre, 2002
- [11] COLLABORATOR (COLLABORative fRAMework for remoTe and mObile useRs) IST-2000-30045. Available at <http://www.ist-COLLABORATOR.net>