Collaborator – Enabling Enterprise Collaboration through Agents

Federico Bergenti[1], Marco Mari[1] and Mercedes Garijo[2]

[1] *Dipartimento di Ingegneria dell'Informazione*
Università degli Studi di Parma
Parco Area delle Scienze 181/A, 43100 Parma, Italy
{bergenti, marco.mari}@ce.unipr.it

[2] *Dep. Ingeniería de Sistemas Telemáticos*
Universidad Politécnica de Madrid
Ciudad Universitaria, 28040 Madrid, España
mga@dit.upm.es

Abstract

This paper presents a particular aspect of the architecture of Collaborator, a software system that provides enterprise users with a shared workspace to support the activities of virtual teams. Collaborator exploits the seamless integration of everyday Web technologies with agents to promote flexibility in the interaction between users. In this paper we concentrate on the agent subsystem of Collaborator, and we show the architecture of this multiagent system and the functionality that it provides. In particular, we present the roles that agenst play in the overall system (personal agents and session manager agent), and we detail on how personal agents are used to (i) *manage the calendar of users;* (ii) *negotiate and classify meetings; and* (iii) *learn the preferences of users.*

1. Introduction

New forms of work in the enterprise rely on the possibility of accessing any resource of the enterprise from everywhere, at anytime. This allows a permanent connection between the employees and the enterprise itself, and it promotes the efficient use of available resources. While the ubiquitous and prompt access to (non-human) resources is a must for a modern enterprise, nowadays it is no more sufficient because the common practice of collaborative work pushes the urge of accessing colleagues as well as resources. New means of communication are needed to enable the coherent and efficient access to all these resources (whether human or not).

From a technological point of view, the Web is assuming a central role in this shift of perspective in the way people collaborate and share information in local and geographic areas [3][8]. The Web is accessible from everywhere and it can integrate different services into a common, easily accessible, platform-independent user interface. For this reason, it has already been adopted as one of the major media for supporting remote collaboration among people [2][6].

This almost complete adoption has been recently exacerbated by the widespread deployment of new mobile networks and by the migration of Web technologies toward them, thus allowing the access of consolidated Web services and facilities from mobile users.

Nowadays, we have the concrete possibility of providing users with integrated and efficient services capable of exploiting the possibilities of both wired and wireless networks to create effective enterprise-wide virtual teams. Project *Collaborator* (*Collaborative Framework for Remote and Mobile Users*) was funded by the European Commission within the 5th Framework Programme in the attempt to validate available and forthcoming technologies capable of concretizing this possibility. The result of project Collaborator is an advanced prototype of a system that provides enterprise users with an efficient and effective means for accessing all resources (whether human or not) available in the enterprise, from everywhere, at anytime.

Besides the design principle of exploiting (standard) Web technologies for Collaborator, the basic communication mechanism that the Web offers is not sufficient to support interactive collaboration. The basic need that stimulated the development of the Web was about consulting structured documents and had nothing to do with supporting an interactive discussion within a virtual team. This is why, project Collaborator decided to exploit the sociability of agents (like, e.g., in [5][10]) to mediate interaction between users of virtual teams.

This paper shows the role of agents in the architecture of Collaborator, and presents the functionality that they implement. In particular, next section details on Collabo-

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

rator and its architecture. Section 3 gives some implementation notes and discusses the main agent-related features of Collaborator. Finally, section 4 draws some conclusions and outlines some future work.

2. Collaborator

Project Collaborator (see the official site, *http://www.ist-collaborator.net*) started on the 1st of November 2001, for an overall duration of two years and involved sever European-wide partners: Space Hellas (project coordinator, Greece), Consorzio Nazionale Interuniversitario per le Telecomunicazioni (scientific coordinator, Italy), Atos Origin Integration (France), Centre Scientifique et Technique du Bâtiment (France), Tecnologia Automazione Uomo (Italy), Technical University of Madrid (Spaing) and Telecom Italia Lab (Italy).

The major goal of the project is the realization of a software system, namely, Collaborator, capable of providing modern enterprises with a shared workspace supporting the activities of virtual teams. Another goal of the project is to set up a trial environment to explore and validate the benefits of integrating Collaborator with emerging technologies, e.g., third generation of mobile networks and terminals, and new generation of home appliances.

Summarizing, Collaborator is intended to support remote and collaborative work in virtual teams meeting the following constraints:

1. *Platform-independence and Web integration*: Collaborator is based on (standard) Web technologies (Java, HTML, TCP/IP, etc.) and it is operating-system and network independent;
2. *Ubiquitous accessibility*: Collaborator can be accessed seamlessly from desktop computers and from handy devices (with sufficient processing power and available bandwidth);
3. *Adaptability to network bandwidth and terminal capabilities*: ubiquitous accessibility requires Collaborator to adapt to the capabilities of terminals and connections that users may access during a virtual meeting;
4. *Flexible user interaction through agents*: people involved in a virtual meeting are associated with personal agents that mediate their interactions and provide profile-based customizations.

The result of Collaborator project is a prototype of a system that respects all these constraints. Such a system is modeled around the idea of *session*.

A session is the view that Collaborator has of a virtual meeting among a number of participants. Such participants use their devices to take part of the meeting, and therefore a session is nothing but a set of (collaborative) activities through a dynamic set of devices that use a number of tools (e.g., applications and document editor) in the scope of a virtual meeting.

A user can join and leave a session at any time, from everywhere. Multiple sessions can run in parallel and a user may participate to more than one session.

A session can be: initiated, ongoing or terminated. In Collaborator, we assume that a session can be initiated and terminated only by a particular user, called *coordinator* of the session, that has full privileges in the scope of the session.

Usually, two or more persons are involved in a session; however, sometime, only one person is participating to a session, e.g., to perform administrative or coordination tasks, like organization of another session.

All users involved in a session share the tools that the coordinator of that session decided to make available. Users jointly exploit such tools (e.g., applications and document editors) to bring about the aims of the session. Each user may access a session from a different terminal, and Collaborator transparently adapts the users' experiences to their contexts, i.e., to the capabilities of their terminals and to the available bandwidth that they can access. Figure 1 shows a running session that comprises editing of a document with Microsoft Word. Desktop users and PDA users have a different view of the shared document being edited.

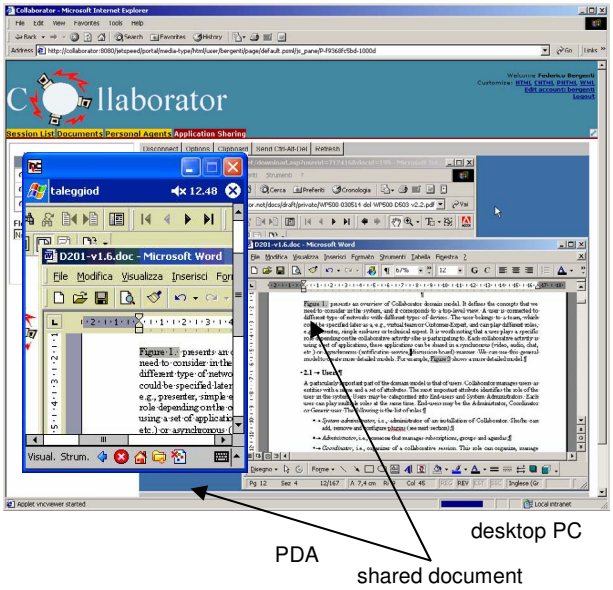

Figure 1. Different views of the same shared tool to users accessing Collaborator through different devices

The architecture of Collaborator is a classic three-tier architecture and it comprises a set of backend server components and one Web client (browser) deployed on each device. Backend components are divided into three categories:

1. *Framework components*: dedicated to manage users, sessions, resources and service components. Parts of the framework are the Session Manager Subsystem, the Resource Repository Manager, Personal Agents and Session Manager Agents;
2. *Service components*: dedicated to provide support to all collaborative activities. Between such service components, the most remarkable are: the Application Sharing service, the Audio/Video Conference service and the Instant Messaging service;
3. *Portal components*: dedicated to provide the tailored output to Web clients. These components are integrated in the framework that Jakarta Jetspeed (see *http://jakarta.apache.org/jetspeed*) provides. This is a portal engine that acts as the central hub where information from multiple sources is customized and made available to different terminals.

3. The Agent Subsystem

Collaborator is designed to accommodate users with different requirements, profiles and devices. Such users need to interact with each other and with the system for preparing sessions, negotiating dates of sessions and managing their personal calendars. All these activities are performed in Collaborator through the use of agents. The *agent subsystem* of Collaborator is based on JADE [1] (see *http://jade.cselt.it*) because such a framework greatly simplifies the implementation of multiagent systems that complies with the FIPA specifications.

More in detail, JADE is not used alone, but Collaborator exploits the possibility of realizing reasoning agents tightly integrated with JADE APIs by means of JESS (see *http://herzberg.ca.sandia.gov/jess*). With this approach, JADE is used to provide the basic structure of the agent and to guarantee FIPA compliance, while JESS acts as the reasoning engine of the agent. JESS supports the development of rule-based systems tightly coupled to code written in Java, and it allows the creation of knowledge bases with information provided in the form of declarative rules.

In order to accommodate agents in the architecture of Collaborator, we decided to use BlueJade [4] (see *http://sourceforge.net/projects/bluejade*), an extension of JADE that can be deployed as a native service of JBoss (see *http://www.jboss.org*). This configuration delegates the management of the lifecycle of the agent platform to JBoss with some advantage in terms of fault-tolerance and scalability.

The agent subsystem is tightly coupled with two important framework components of the architecture of Collaborator:

1. The Session Manager Subsystem (SMS), that activates Session Manager Agents (SMA) when needed, i.e., when negotiation of a session is requested;
2. The Resource Repository Manager (RRM), that stores users' profiles and personal calendar for the agent subsystem.

Figure 2 shows the architecture of the agent subsystem, and the rest of this section details the elements that it comprises.

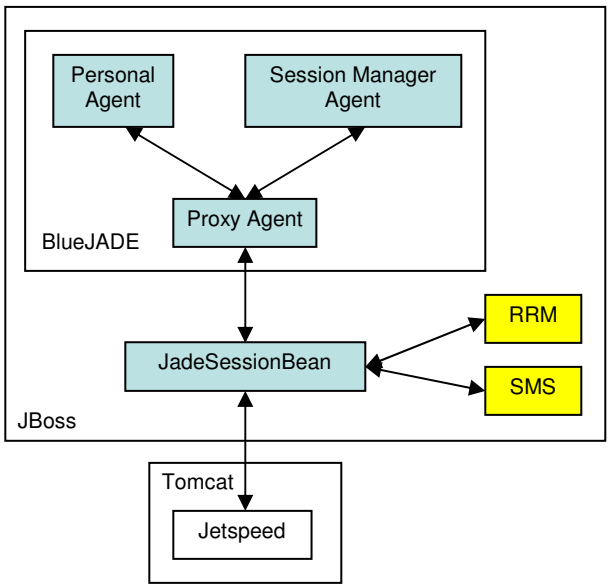

Figure 2. Architecture of the agent subsystem, and its relation with the rest of the architecture of Collaborator interface

3.1. Personal Agents

A Personal Agent (PA) is in charge of working on behalf of a user. In particular, a PA has the following duties:

1. *Manage the users' personal calendars*. Users can use their PAs to view/modify their personal calendars (add, edit and cancel meetings), and to import/export meetings and contacts to/from their personal calendars from/to other calendar applications. Such import/export functionality exploits common vCalendar (see *http://www.imc.org/pdi*) file format, that allows a seamless integration with popular calendar applica-

tions like Microsoft Outlook.

2. *Negotiate session on behalf of its user*, i.e., negotiate date and time of sessions using the information that PAs have on the users' calendars and time preferences. PAs are able to send messages to the user via e-mail to inform them about new virtual meetings that they negotiated.
3. *Classify meetings* (sessions and personal meetings) in user-defined categories.
4. *Learn time preferences of a user*. PAs generate the information needed to describe their users' time preferences when fixing a meeting.

All the information needed to accomplish all these tasks is retrieved and stored by PAs through just-in-time calls to the RRM when any change occurs.

Figure 3 shows the interface the PAs provide to users for managing negotiated and non-negotiated meetings.

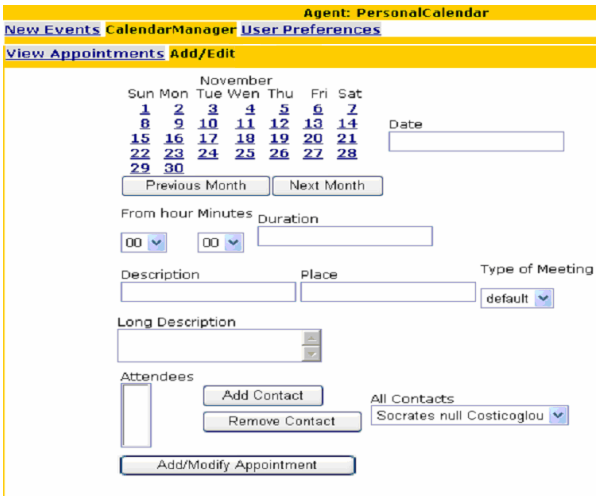

Figure 3. Interface of Personal Agents for managing meetings (detail)

Negotiation. The negotiation process of a session is intended to maximize the number of participants, taking into account their time preferences. It is carried out by the PAs and the SMA created by the SMS for the negotiating session. The SMA asks to the invitees' PAs the time preferences of their users for that particular session. Each PA classifies the session in one of its user-defined categories and, on the basis of this classification, each PA sends to the SMA its user's time preferences. With all this information, the SMA finds a suitable date and asks the PAs to confirm it. If all PAs confirm the date, the process ends. In the case that some PA cancels its participation, the negotiation process is restarted excluding all the solutions found in previous processes.

Classification. In order to classify a meeting, whether collaborative or personal, in a user-defined category, a PA creates a document vector and calculates the cosine distance [9] between this vector and the vectors representing the categories defined by the user. The shortest distance indicates the user-defined category for classifying that particular meeting. This simple classification algorithm requires a user defining her categories. This is accomplished by giving her PA the name of the new user-defined category and:

1. the meetings that it should contain; or
2. a set of words that it should contain.

With this information, the PA is able to generate a document vector that includes all the words explicitly given by the user, or contained in the meetings given by the user. For a session (i.e., for a meeting managed by Collaborator), these words include *(i)* the participants, *(ii)* the required expertise needed to attend to the session and *(iii)* all the words contained in the sessions' short and long descriptions.

When a meeting is classified under a certain user-defined category, all words that are not already included in that category are incorporated in it. The user is also able to change the category of a meeting. When this happens all words of that meeting are deleted from the old category and included in the new one. If a particular meeting cannot be classified in one of the user-defined categories, then it is classified under a predefined default category.

Learning. Meetings are distributed all along the user's personal calendar and classified in user-defined categories. PAs use an ID3 learning algorithm [7] for supervised classification to generate rules that associate user-defined categories with time parameters (e.g., *"the user prefers to attend brainstorming meetings late in the evening"*). PA are able to learn for the following time parameters: hour of the day, day of the week (Monday, Tuesday, etc.), month of the year (January, February, etc.) and day slots (morning, evening, night, etc.).

All rules learned by a PA have a degree of confidence. Such a degree grows if the rule leads to correct results and it decreases if the rule leads to erroneous results. To manage this degree and to complete the supervised classification, PAs also have a feedback mechanism. When the user introduces a meeting in her personal calendar without negotiation, the PA autonomously calculates three possible dates for that meeting. If such estimated dates are close to the ones directly selected by the user, the degree of confidence on the rules that led to the correct result is raised. If such estimated dates are different from the ones directly selected by the user, the degree of confidence in

the rules that led to the erroneous result is lowered.

The user also has the possibility to directly give rules to the PA (e.g., *"I do not like meetings after 5pm"*). In this case the rules' degree of confidence is the highest possible, and the feedback mechanism described above does not apply.

All rules obtained from the ID3 algorithm are stored in the RRM and they are loaded in the rule-based engine to obtain a user's time preferences. Such rules are implemented as a set of Javabeans that manage the degree of confidence of each rule.

The user's profile that PAs manage comprises the user-defined categories and the rules that describe the user's time preferences.

3.2. Session Manager Agents

Session Manager Agents (SMAs) are in charge of negotiating and managing sessions and the set of users attending such sessions.

In order to create a new session, the SMS creates a new SMA and gives it the chairman privileges for that session, so to make it choose the range of dates, the range of hours and the invitees. The newly created SMA negotiates with PAs the list of attendees and the best suitable date for the session. Once a session is completely negotiated, the SMA stores such information in the RRM.

Another duty of the SMA is to inform users with an e-mail when the chairman cancels a session or when an attendee cancels her participation. In this last case, the attendee can propose a substitute. The SMA negotiates with the PA of the substitute her availability for that session.

There are as many SMAs as sessions in the system, one per session. In order to reduce the system load, SMAs are destroyed when they finish their work (end of negotiations) and recreated if needed (e.g., to negotiate a substitute).

3.3. Interfacing with Agents

The main purpose of agents in Collaborator is to deal directly with users, e.g., to manage their preferences or to act on their behalf. Nevertheless, other components of Collaborator can use the services that agents provide to give better quality to the functionality that they implement. The interface between agents and the rest of Collaborator is implemented through two elements of the architecture depicted in figure 2, namely JadeSessionBeans and Proxy Agents.

A JadeSessionBean is a stateless session bean offered to the rest of Collaborator to access the agent subsystem. It is basically a façade for the agent subsystem and its lifecycle (and subsequent services) are managed directly by JBoss, thus exploiting the features that it provides in terms of fault-tolerance and scalability. It is available through JBoss JNDI service (context name *"agents/JadeSessionBean"*) and it groups all methods needed to interact with the agents living in BlueJADE.

All agents of the agent subsystem use FIPA ACL to communicate and to interact with each other. Proxy Agents are in charge of routing ACL messages from JadeSessionBeans to the corresponding agents. These messages are used to ask to agents for desired actions or inform them about new system-related facts.

4. Conclusions

This paper describes the role of agents in the architecture of Collaborator and the functionality that they implement. Agents are used to provide concrete advantages to users because they are meant to allow reducing tedious interactions between users, thus maximizing the efficiency of communication. In particular, Personal Agents *(i)* negotiate sessions autonomously on behalf of their users, *(ii)* manage their personal calendars, *(iii)* classify their meetings and *(iv)* learn about their time preferences. Similarly, Session Manager Agents are created with the aim of leading the negotiation of new sessions.

Agents are useful for desktop users, but they become nearly indispensable for mobile users, who have limited bandwidth and limited power autonomy. Mobile users can delegate to their Personal Agents all the organization tasks they cannot easily manage.

The most promising future direction in the use of agents in Collaborator is their application in solving complex tasks related to the management of sessions. In particular, they could be used to automatically distribute the floor or to provide a voting mechanism during running sessions. They could also implement automatic notification mechanisms for users that cannot attend a meeting, or that need to attend more meetings concurrently. The Personal Agent of a user could follow a meeting on her behalf and it could notify any interesting event to her. These and other developments of this research are planned for the near future in the scope of an extension of the Collaborator project or in similar initiatives.

Acknowledgements

Authors acknowledge the contribution made by all members of the Collaborator consortium, without which the work described here would not have been possible.

The European Commission funds this work through the grant IST-2000-30045 (Collaborator).

References

[1] F. Bellifemine, A. Poggi, G. Rimassa. Developing multi-agent systems with a FIPA-compliant agent framework.. *Software Practice and Experience*, 31:103-128, 2001.

[2] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, G. Woetzel. Basic support for cooperative work on the World Wide Web. *International Journal of Human Computer Studies*, 1997.

[3] F. Bergenti, A. Poggi, M. Somacher. A collaborative platform for fixed and mobile networks. *Communications of the ACM*, 45(11):39-44, 2002.

[4] D. Cowan, M. Griss, B. Burg. Bluejade – A service for managing software agents. Technical report, Hewlett-Packard Labs, 2001.

[5] C. Ellis, J. Wainer. Groupware and computer supported cooperative work. In G. Weiss (Ed.) *Multiagent Systems*, MIT Press, 1999.

[6] T. Hodes, R.H. Katz. A document-based framework for Internet application control. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems (USITS '99)*, Boulder, CO, 1999.

[7] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993.

[8] D. Ramduny, A. Dix. Why, What, Where, When: Architectures for cooperative work on the World Wide Web. In H. Thimbley, B. O'Connail, P. Thomas (Eds.) *Proceedings of the Conference on Human Factors in Computing Systems*, pages. 283–301, 1997.

[9] J. R. Smith. *Integrated Spatial and Feature Image System: Retrieval, Analysis and Compression*. PhD thesis, Columbia University, 1997.

[10] Y. Ye, S. Boies, P.Y. Huanf, J.K. Tsotsos. Agent-supported adaptive group awareness: Smart distance and WWW-aware. *IEEE Transaction on Systems, Man and Cybernetics*, 31(5):369-380, 2001.