

A Multiuser Groupware Calendar System based on Agent Tools and Technology

Alberto Grosso, Christian Vecchiola, Mauro Coccoli, and Antonio Boccalatte

Department of Communication, Computer and Systems Sciences, University of Genova,

Genova, Italy 16145

Phone: (+39) 0103532284

Email: {agrosso, christian, coccoli, nino}@dist.unige.it

Abstract — In this paper MATT, a Groupware Calendar System (GCS), is presented. The MATT system provides users with a shared calendar service by preserving the right to the privacy and also by presenting features for arranging meetings automatically. The core of the proposed GCS system is a Multiagent infrastructure which manages the shared calendar and drives the process of arranging meeting in an effective way. An advanced network management system allows MATT working regardless the connection is present or not.

Index Terms— Groupware, Groupware Calendar System, Agents, Multiagent System.

I. INTRODUCTION

HIGH mobility and connectivity are the main stream of novel enterprise applications. The competitive global business environment demands that employees in a company are very mobile and very flexible so that people spend a lot of time with customers, on the road or sitting in airport lounges. New models of business are based on a distributed environment but, on the other hand, they require a high degree of collaboration. In this respect, enterprise portals give broad access to the enterprise information system as being over a world wide extranet as well as distributed applications accessible from everywhere over the Internet give the home feeling. People and workers can have access to their desk, chest, and drawers even from abroad, increasing their productivity as well as enhancing their social life. Indeed corporate intranets hold a wealth of vital information - from calendars and contact lists, to key customer documents and reports - and they provide access to external Internet-based information giving their employees the ability to gain fast and easy access to these shared resources so that being out of the office does not mean being out of reach of the business-critical information and services residing on the corporate intranet.

Technological solutions are provided by ICT in terms of portable devices with networking capabilities as well as with specific collaborative tools. Supported by the spread of the web, the re-raise of platform independent programming languages and tools, object frameworks, software engineering,

and component based solutions; a flexible technological scenario can be depicted. Given such a powerful, full featured technological infrastructure, a variety of new services can be considered, working regardless they have a network connection or not, as long as they will have one sometimes. In fact office automation tools and services can support workgroup and collaboration through specific replication and synchronization mechanism enabling applications to work both on-line and off-line according to business trends like downsizing, flexible manufacturing systems, and just in time production.

Mobile professionals can access the information they need over the intranet or the wide world of the Internet, no matter where they are. A long way from the office, right in the middle of a business deal, they can check warehouse stocks, place an order or find out where a shipment is. And then close the deal. Moreover they can also check their e-mail, handle files on the intranet, and take care of messages. Furthermore, the Enterprise information system can supply gateway/proxy services allowing the 'pushing' of information such as stock quotes, calendar events, and news alerts to users, so the latest information is instantly available. This may be independent of permanent Internet connections. This can result in an improved productivity and performance for the organization, and an enhanced competitive edge.

Recalling the three types of enterprise applications: intra-organizational, inter-organizational, customer-to-business [1], one can observe that a huge number of software systems have been developed in support of the above applications. Due to the distributed nature of the enterprise system from both the organizational and the geographical point of view, distributed computing is a corner stone in enterprise information systems and technology. The client-server model (based on message passing and RPC) has been the dominant paradigm for many years. Recently, the client-middleware-server model has grown in popularity thanks to the introduction of middleware components such as CORBA [2] and DCOM [3]. In the same time, agent technology has been gaining popularity in the scientific community and much literature exists on agents and multi-agent systems as well as frameworks and development tools.

Agent technology is well suited for the design of distributed and concurrent applications requiring a high degree of cooperation with asynchronous communication; hence agents represent an effective solution for designing and implementing groupware systems.

In this paper a collaborative tool for enterprise solutions is presented that is a groupware calendar system based on AgentService framework. The project is named MATT, acronym for both "Multiuser GCS based on Agent Technology and Tools" and "Monday At Ten Thirty", a friendly name recalling the basic functionality of MATT itself. The choice is up to the readers which may feel more comfortable with the first or the second name.

After this short introduction, a survey of groupware calendar systems is presented in Section 2. Agents and Multiagent technology are discussed in Section 3. Section 4 describes the MATT system introducing its features and focusing on distributed architecture and the negotiation protocol. Conclusions follow.

II. GROUPWARE AND COLLABORATIVE SYSTEMS

Groupware is the technology designed to facilitate people working together, collectively, while located remotely from each other. Ellis defines groupware as "computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment." [4].

Generally speaking, most groupware applications are characterized by some common aspects: communication strategies (synchronous and asynchronous), coordination (organization of users interactions), and distribution (referring to the physical location of the group member); scalability and flexibility are to be considered too. Groupware services can include the sharing of calendars, collective writing, e-mail handling, shared database access, electronic meetings with each person able to see and display information to others, and other activities such as replication of files across a distributed system so that all users can view and share exactly the same information. The support of dynamic collaboration is the basic requirement for future enterprise systems and groupware technology may be used in this context to communicate, cooperate, coordinate, solve problems, compete or negotiate. Groupware offers significant advantages over single-user systems. The use of collaborative technologies facilitates communication forming groups with common interests and it brings together multiple perspectives and expertise.

Sometimes called collaborative software, groupware is an integral component of a field of study known as Computer-Supported Cooperative Work or CSCW. According to Wilson, CSCW is a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services, and techniques [5]. Key issues of CSCW are group awareness, multi-user interfaces,

concurrency control, communication and coordination within the group, shared information space, and the support of a heterogeneous, open environment which integrates existing single-user applications.

Groupware and CSCW offer multiple perspectives of the enterprise system, they deal with technology issues, due to distribution flexibility and interoperation among software artifacts; they face social and organizational problems rising from interpersonal communication that should take care of both individual and company point of views [6].

Since the very beginning Groupware Calendar Systems (GCSs) were developed in the field of groupware application (e.g. white 77 [7]). A GCS is a system managing calendars that can be shared across a network. GCS creates new opportunities for social coordination, conflict resolution, and challenge notions of personal privacy and control over information and time. GCS provides a common place where the personal calendar of every single user is shared and can be accessed on-line with different confidentiality policy. Collaboration is commonly supported by simply sharing or by viewing other people's calendars, or by sending special meeting invitations through the GCS. A higher level of collaboration can be reached in order to make automatic the scheduling of meetings and to offer features for fixing time and date in arranging a meeting.

III. AGENTS AND MULTIAGENT SYSTEMS

The adoption of agent and multiagent technology is a challenging approach for modeling and implementing groupware applications and tools as well as collaborative systems in general. A place where agents interact, compete, cooperate, and negotiate in order to achieve their goals is an effective abstraction model for describing a collaborative environment.

Generally speaking, in groupware applications there are not standard rules and clear policies, due to the fact that most decisions are taken by people so that the system heavily depends on a human factor. Because of their nature agents can be thought as exhibiting human-like behaviour. The concepts usually applied to human thinking can also be applied to agents that are, by the definition, able to take autonomous decisions, to take the initiative, and to communicate with peers. Social ability can be profitably exploited in groupware applications where many actors, roles, and functionalities are involved; hence a MAS is a suitable infrastructure for such solutions.

This section covers the main aspects of agent oriented paradigm from a theoretical point of view and for this specific application as well. The AgentService framework used for developing the system is presented.

A. Agents

An agent is an autonomous software entity with some level of "intelligence" [8] that makes it act in a human-like fashion: agents are socially able, they react to environmental changes,

and they take the initiative to pursue their goals. While a complete definition of software agent is still an open issue [9], some qualities that characterize this concept are widely accepted by the community of researchers. Agents are situated in an environment and interact with it in order to meet their design objectives. By means of cooperation, negotiation, and competition agents interact with peers to enhance their performance. These qualities characterize the agent's social ability. Furthermore, pro-activity and reactivity make the agents exhibit a flexible and autonomous behaviour. Pro-activity is the ability to show goal driven behaviours by taking the initiative, while reactivity is the ability to cope with the environmental changes in a timely fashion. Pro-activity and reactivity lead agents to exhibit a human like behaviour. These features define agents as more complex entities than objects: they have control over their behaviour (not only over their state); agents interact through a set of messages that define a conversation (not by method invocation). Nevertheless, they are commonly implemented as autonomous, concurrent, and self-contained objects.

B. Multiagent Systems

Agents are designed to operate in a community. Such a community is normally called a multiagent system (MAS) [10]. MASs are decentralized systems with distributed control and asynchronous computation, they provide a runtime environment and constitute the infrastructure for the agent interaction and communication. MASs provide a set of facilities to agents such as the message transport system and directory services. Such services are provided by an agent platform that is the physical counterpart of the multiagent system. In the community of researchers and software engineers many agent programming platforms were delivered, each of them showing different features. An abstract specification of the architecture for an agent platform has been proposed by the Foundation of Intelligent Physical Agents (FIPA) [11], an international organization promoting standards for agent technologies. The architectural model proposed by FIPA is commonly taken as a reference in making comparisons between different implementations of agent platforms.

The model proposed by FIPA includes the presence of some components that are considered compulsory for every implementation of a multiagent system. These components are the following ones:

- *Agent Management System (AMS)*. AMS is responsible for the life-cycle of the agents, schedules their activities, and maintains a registry of all the agents hosted into the platform. By using this registry AMS offers a *white pages* service to the agents and also to the platform users.
- *Directory Facilitator (DF)*. DF maintains a *yellow pages* service accessible to all the agents hosted into the platform and to the platform users. There can be more than one directory facilitators in one platform and they can constitute a federation that goes beyond the boundaries of a single agent platform.

- *Message Transport System (MTS)*. MTS represents the interface to the messaging system of the platform. Agents interact with the MTS in order to send messages to other peers either if they are on the same platform or on different platforms.

These components can be either agents or components implemented with any other model. The FIPA reference model does not comprise the internal structure of agents but focuses the attention on the interoperability between different agent platforms by defining the core services of each platform for interoperation and an agent communication language (ACL) that every agent should be able to understand.

C. Agents and Collaborative Systems

Cooperation, competition, negotiation, and social skills in general are fundamental aspects of either the groupware philosophy or its implementation in software systems. The use of abstractions that provide elements naturally offering these features, like the agent technology, could be an interesting approach to design and to implement groupware solutions. The ability to interact with peers in a community (like agents in a multiagent system do) makes the agents a winning solution for this kind of applications. Groupware applications are communication-centered and agents perform their activities by using the communication as a basis for interaction.

Agent oriented programming offers an abstraction level very useful either in the design phase or in the implementation phase: agents have by design many of the features that a groupware system requires. In particular, agent oriented technology can be very useful to model group calendars: agents act like avatars of the people they represent and are able to perform many tasks without the intervention of their owners. Smart software assistants should have a reasonable degree of autonomy in order to accomplish their tasks and this is a feature that agents have by design. The ability to take the initiative becomes crucial when it is necessary to schedule an appointment in a different date or when a request for a new meeting has been notified: agents are proactive and can manage these tasks without direct user intervention. Moreover, cooperation is fundamental in this scenario: scheduling a meeting requires negotiation and cooperation between the different assistants. Agents are socially able by design and can coordinate themselves to find a feasible schedule by also taking in account the personal needs of their owners. Agents are certainly an effective abstract model to cope with the requirements for the management of a group calendar.

Agent technology is also a valuable solution to implement the software infrastructure for a group calendar: multiagent systems offer services and an infrastructure supporting the activity of agents. Agents expose a message driven interaction and the scenario devised by a group calendar is message oriented: multiagent systems provide a communication infrastructure that constitutes the natural communication channel for each agent. Software applications designed for

group calendars need to maintain and to manage a registry of the users; multiagent systems provide a registry service where every agent can be found either by its name or by its capabilities. It is worth to notice that in this case there is no need to design a software component that takes care of the registry, but each agent has only to register itself with the MAS and query it to find what it needs. Finally, group calendars are dynamic: new users can join the group when new projects starts and some of them are removed when the collaboration terminates. MASs are designed to seamlessly manage the introduction of new agents as well as their removal. This feature is accessible without any additional cost and without any impact on the preexisting community of agents: the white and yellow pages services make agents aware of every change that should happen in the community of agents.

By using agent technology software engineers can design systems more quickly and by focusing their attention on the specific issues related to the system rather than on the technical aspects of implementation of the infrastructure. Moreover software agents constitute an easy to understand interface especially in the case of group calendars where the interaction with a personal assistant resembles the one that managers have with their personal assistants.

D. AgentService

AgentService [12] is an agent programming framework built on the top of the Common Language Infrastructure [13]. The framework provides the programmers with the following features:

- definition of autonomous, independent, and persistent agents;
- concurrent execution of agents and their multi-behavior activity;
- persistent shared data structures within a single agent;
- transactional agent communication based on message exchange;
- access to the FIPA service components (AMS, DF, MTS).

One of the key features of AgentService is its agent model that is not particularly tied to specific agent architectures but flexible enough to implement different ones. Within AgentService an agent is constituted by a set of knowledge objects and a set of behaviour objects. Knowledge objects define the agent's knowledge base while behaviour objects define the activities that an agent can perform and the services it offers to the others. Knowledge objects are shared among the different behaviours that are scheduled in a concurrent manner. The definition of a new type of agent, leads to the definition of a template which specifies the behaviour and knowledge objects that characterize it and to the definition of these behaviour and knowledge types.

AgentService also provides a set of extensions to the C# language called Agent Programming eXtensions (APX) [14] and specifically designed to ease the development. APX are a set of templates modeling the implementation of agents, behaviours and knowledge objects, represented as types in the

programming language. The extensions also include a tailored compiler that automatically applies the programming patterns required by the AgentService framework. The use of APX simplifies the development activity and helps the programmers to follow the AgentService programming guidelines by detecting at compile time the erroneous programming patterns, if any are used.

IV. MATT ARCHITECTURE

A. Introduction

The design of a GCS system involves, as most groupware applications, three different fundamental perspectives: personal, social, and technological. First of all it is important to keep in account requirements that a single user should desire within an easy to use shared calendar. To this aim let us consider a reminder, facilities for planning and scheduling meetings, and a customizable user interface with friendly programmability features. Since GCS is designed for users working in a company or in a team, the social aspect assumes big relevance in the design process. The GCS system has to exploit the advantages of a shared calendar to inform working groups and teams about scheduled events and personal tasks time and date. All these functionalities must respect the right to the privacy of all users, making public the availability of a person while preserving the subject of the appointment, if tagged as "confidential". The last topic involved is the technology infrastructure which should provide a reliable, distributed, and asynchronous environment. The system is distributed due both to topological matters of the company (buildings may be scattered on a wide geographic area) and to the mobility gained by the workers in the global market. The communication among users in the GCS has to be asynchronous for the physical connection may be or not always on, depending on many factors. According to these guidelines, the MATT system is a GCS designed for working in an Internet based environment, which can be accessed from everywhere with any web enabled device.

In addition to the basic functionalities that any GCS systems offer, MATT relies on a multi-agent system that makes it smarter than others. Each user relies on an avatar in the multi-agent system; avatars interact and cooperate by sharing information related to their own calendars and tasks. Information is made available to the web by means of web services which make the system accessible through any standard web device. On the other hand, from the client-side, users can have a rich user interface as well as a lightweight application for accessing data on the server over the web. Depending on the specific client (laptop, portable device or smart phone), the server can manage the connection and give appropriate results. Agents have a leading role in the user interface too: a local agent is running on the client, managing the calendar system and the whole data related to the user activity. The local agent is simply a replica of its counterpart on the server, hosted in the agent platform. Whenever the

network connection is not available, the local agent acts on local data maintaining full functionalities to the GCS. In a few words, while the client is off-line, the local agent just notifies forthcoming tasks and meetings, if any, while updating the calendar according to the user instructions. When the connection is restored the server is contacted by means of a synchronization mechanism for maintaining information up to date, addressing consistency too. Should new appointments overlap (maybe they are registered separately on the client off-line and on the server) an automatic negotiation is started in order to reach a common agreement, always based on the rules included within the agent platform driving the whole system.

B. Hierarchical Model

Within MATT a hierarchical model is introduced, based on the concept of *units*. In this respect a unit is the fundamental building block in the hierarchy tree representing a social structure. Single units have connections and dependencies with each other and a relevant set of rules has to be defined to specify interactions, ranking, and roles. In this GCS application, two classes of units are defined: positions and organizational units. Positions are attributes that can be ascribed to users while organizational units are divided into teams, sections, and organizations. Organizations include sections that can contain teams and teams are made up of people identified by positions. MATT specifies neither the nature of these units nor the different types of positions in order to be flexible enough to model a variety of social groups. For example the social structure within a University can be easily represented by matching the different research groups to teams, the departments to the sections, and the faculties to the organizations. People are described by the positions they cover: in this particular case they can be professors, researchers, graduate and undergraduate students, or administrative personnel. On the other hand, a group calendar handling the schedule of sport events, matches, and the communication among team players, would only define positions and teams, and the would map the league to which teams belong to a section contained in a default organization.

The hierarchical structure also defines the capabilities of each unit. Organizational units containing other organizational units can issue meetings and appointments that have a higher degree of priority than the ones issued by the contained organizational units. Moreover, additional rules can be defined in order to model relations that go beyond the hierarchical structure defined above: these rules have a priority higher than the ones that are inferred from the hierarchical structure.

The hierarchical model specified by MATT can be defined by a tuple $H \langle P, T, S, O, R, M \rangle$ where:

- P is the set of positions;
- T is the set of teams;
- S is the set of sections;
- O is the set of organizations;
- M is a map that defines the structure of the hierarchical model;

- R is the set of relations that are added to the map.

A role R is defined by the triple $\langle Id, Pos, O \rangle$ where Id represents the identifier of the user, Pos is the position the user holds, and O is the organizational unit in which the user holds that position. Roles clearly identify the position and the capabilities of users into the hierarchy. Capabilities are fundamental when agents have to schedule meetings and need to solve conflicts originated by appointments that are requested in overlapping time spans. An appointment is defined by a tuple $A \langle N, L, R, I, Pr \rangle$ where:

- N is an identifier for the appointment;
- L is a triple $\langle D, S, Pl \rangle$ where D is the date of the appointment, S (span) is the duration, and Pl (place) an identifier of the location;
- R is the role of the person who asked for the appointment;
- I is the set of people invited described by their identifiers;
- Pr defines the priority level of the appointment.

The role of the requestor and the priority level of the appointment give all the information to the other participants to reject or to accept the appointment. Each agent involved in the distributed negotiated protocol described in section 3.H, will look at the date of the appointment and in case the time span is overlapping with a previous appointment it will compare the priority level. If the priority levels are the same the role of the requestor will determine which appointment need to be discarded while in the other cases the appointment with the lowest priority level will be scheduled again.

C. Architecture

Figure 1 describes the structure of MATT that follows the well known three tier model. On the client side smart clients like PDAs, rich web clients plus web services, or simply web browsers are used to look-up the calendar, to register new appointments, and to be notified of incoming meetings. Different types of clients have been designed, simple web browsers give limited interaction with the agenda while desktop applications are able to work off-line and synchronize the data while connected. On the server side, the web application gives access to the multiagent system managing the group calendar. The multiagent system uses a RDBMS to store all the data related to the user profiles, the calendar structure, and the appointments. All the features offered by MAT are accessible through web service technology. This is a key point of the entire architecture: smart clients will directly connect to these services while web browsers will navigate the web site hosted into the application server to perform all the operations. The application server will use the web services on behalf of the users connected and will interact with the multiagent system.

D. Multiagent System

The multiagent system hosts the community of agents managing the group calendar and is implemented by using one or more instances of the AgentService platforms. AgentService provides the agents with all the needed services and gives access to the MAS through web services.

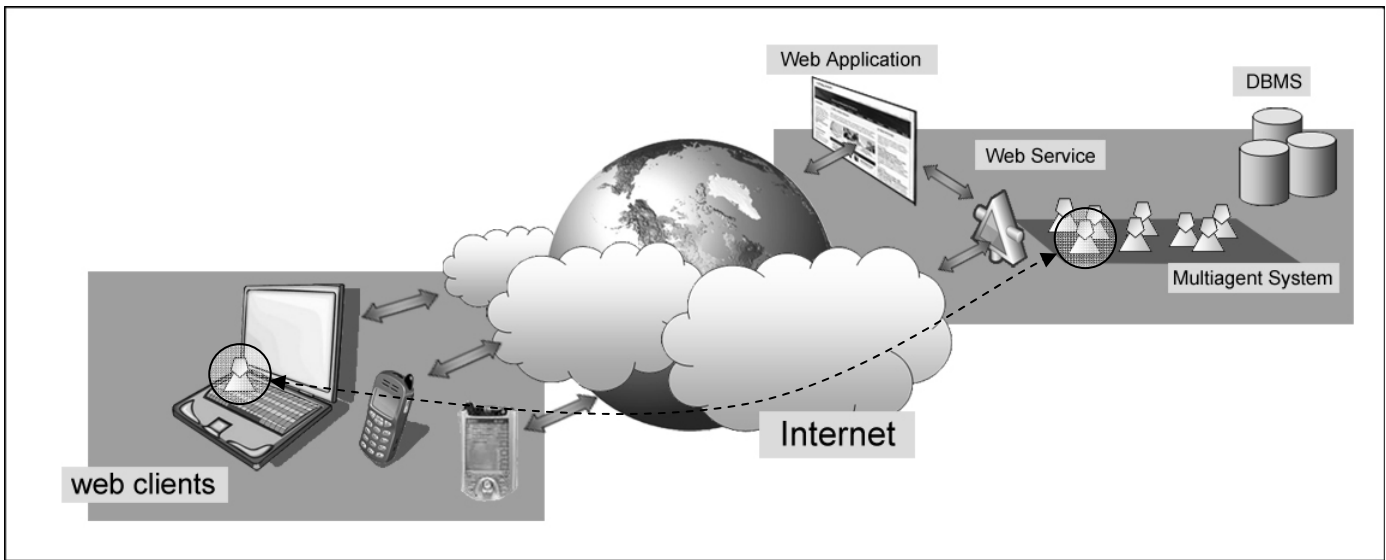


Figure 1. MAT Architecture

Two types of agents have been designed to implement the system defined by MATT: personal assistant agents and hierarchy manager agents. Personal assistant agents take care of the user's personal agenda while service agents manage the set of relations among the people sharing their data with the others.

Inside the MAS there is one Personal Assistant agent for each user of the group calendar taking care of:

- registering new appointments decided by the owner into the personal agenda;
- handling incoming requests of new meetings;
- notifying its owner of the incoming meetings;
- contracting with all the involved agents in order to find a feasible schedule for a meeting composed by many people;
- synchronizing the agenda with the user's agenda locally stored, if any.

All these tasks are performed by maintaining the owner's privacy on the nature of its appointments.

The agents exploit the platform services in order to interact with peers. Each agent registers itself to the AMS, to the DF, and to the MTS. Personal assistants can be found by querying the AMS, while DF is useful when it is necessary to find out all the people belonging to a group or to an organization. This feature is particularly useful when one personal assistant has to set up a group meeting, while the AMS is queried when one user needs to meet specific users whose names are known. All the interactions required to schedule meetings and set up appointments are performed through a message oriented protocol and the MTS takes care of the delivery of all the messages regardless they are on the same platform or on a different platform.

The agent template characterizing the personal assistant agents is made up of knowledge units and behaviours as required by the AgentService framework (see section 2.C).

Figure 2 illustrates the relations among knowledge units and behaviours.

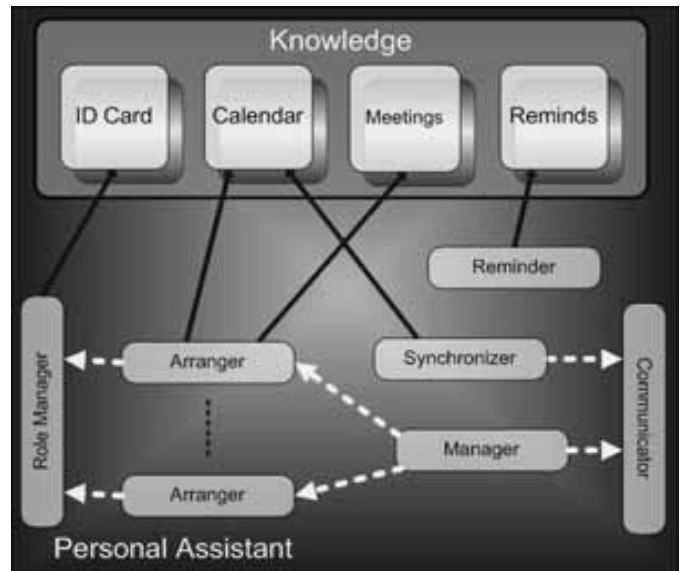


Figure 2. Internal structure of the Personal Assistant Agent

The set of knowledge units describing the agent's knowledge base are the following:

- **Calendar:** it represents the user's agenda split in time slots. Each appointment is defined by the tuple A (see section 3.B); by accessing this knowledge unit the agent knows all the planned tasks indicating the people involved and their priority level.
- **ID Card:** it defines the identity card of the agent; it provides information about agent unique identifier (AID), access credentials, and the role belonging to the user the agent represents.
- **Reminds:** it contains appointments or notices for which

the user wants to be alerted; it is formed by a set of alarms events that will fire on a scheduled time slot and notes that needs to remembered.

- *Meetings*: it contains all the meetings a personal assistant has to arrange or for which is requested the owner's presence along with the time and the date. This is the most important knowledge unit in the meeting schedule process performed by the arranger behaviour.

Each personal assistant agent is characterized by four different types of behaviour objects:

- *Communicator*: it performs all the communication activity among personal assistant agents: meeting requests, agreements, and rejections. Communicator is also involved in the synchronization process between client and server agents.
- *Arranger(s)*: it implements a specific algorithm for the interaction protocol defined to schedule meetings preventing inconsistency in the calendar.
- *Manager*: it instantiates and manages the arrangers behaviours when a new meetings has to be scheduled in the calendar.
- *Role Manager*: it determines the importance of a meeting analyzing the position of the requestor in relation to the hierarchical structure. It provides to Arrangers the basic information to discriminate among appointment priorities driving their choices.
- *Reminder*: it alerts the user about the events scheduled in the Reminds knowledge units.
- *Synchronizer*: it interacts with the corresponding client agent in order to synchronize its calendar with the one managed by remote agent in the client side.

The Client Personal Assistant agent is modeled as its server counterpart described above but it does not include the appointment scheduling functionality. For this reason Client Personal Assistants are not provided with Arrangers, Role Manager, and with the Meetings knowledge unit.

Hierarchy Manager agents take care of setting up the system by creating all the data structures in the RDBMS in order to manage the social structure of the users sharing their calendars. While the system is running they make MATT administrators able to modify the hierarchical model. Hierarchy Manager agents operate on the data model updating it according to changes in the organizational structure: new teams may be created for a new project, new users may be added to the system, changes to the organizational units relation can be done. The template defining a Hierarchy Master agent contains a Rules Knowledge Unit to hold the new rules for the hierarchical model and an Update Structure Behaviour to introduce these rules (adding users, remove teams, etc.) in the organizational structure.

E. Database Management System

The RDBMS stores the data about the organizational structure of the group calendar users. When the system is setting up service agents create a new database and define a set of tables that will store:

- roles, positions, teams, sections and organizations that are involved in the group calendar;
- relations that hold among the different organizational units that do not follow the hierarchical structure;
- users profiles;
- users personal agenda.

The tables and the relations defined in the RDBMS are set up, maintained, and modified the by the Hierarchy Manager agents. The RDBMS is queried by the Role Manager behaviour of the personal assistant agents in order to retrieve all the data about the requestor of a meeting and about its capabilities. When personal assistants need to ask for a group meeting they look up the RDBMS in order to find out all the users belonging to a team, a section, or an organization.

F. Web Application

The web application hosts a web service that constitutes the access point to the system: every type of client interacts with the web services in order to use the features of MATT. Web services guarantee a wide access to the system through a standard communication channel.

The functionalities offered by MATT cover either the administrative part of the system or the services designed for the group calendar users. The administrative part includes the following:

- insertion, deletion, modification of units into the hierarchy;
- insertion, deletion, modification of relations into the hierarchy;
- management of roles.

By using the web service group calendar users can access their personal calendar, reserve appointments, request group meetings, and synchronize their local calendar with the one maintained by the server. They are also notified in time for incoming meetings.

All the services become available after the users or the administrators authenticate themselves against the system. The Web service provides the clients with a customized environment based on the credentials given. In this way the privacy of each user is maintained and guaranteed.

The web application hosts a web site that allows users to remotely access the group calendar by means of common web browsers. The web site interacts with the system through web services that is the unique access point to the system. Simple web browsers do not allow users to work off-line but they can always look up their calendar, sign-up new appointment and ask for group meetings. Administrators do not have to synchronize data and simple web browsers give them the same functionalities offered by smart clients.

G. Smart Clients

Smart clients provide the users with the feature of working off-line. A smart client is a lightweight application containing a remote software agent that acts as the personal assistant agent of the user in the multiagent system. The features of remote agents are a subset of the ones exposed by the

corresponding personal assistants hosted in the MAS. Remote software agents cannot be involved in the negotiation process used to schedule appointments or meetings: since they are designed to work off-line they cannot interact with the other personal assistants that are involved. While the smart client is connected, the remote software agent can synchronize its local data with the corresponding personal assistants in the MAS. When the client is not connected the local personal assistant can reserve time slots for appointments or group meetings: these requests will be fulfilled when the client will be connected to the system. This aspect is a key point: when the user reconnects to the group calendar, potential conflicts could arise among the appointments scheduled while the user was off-line, and the appointments reserved by user himself in the same period of time spent without network connection. In this case the roles and the priorities related to the appointments are evaluated to solve the conflict and to decide which one of the overlapping tasks has to be scheduled again.

H. Negotiation: Protocol and Rules

In the MATT system the process for arranging a meeting is performed in an automatic way. The user that wants to organize an event simply has to ask to its personal agent to arrange the requested appointment indicating all the correlated information. It is important to specify date, place, and name of the event, the people involved and the level (e.g., important, urgent) of the meeting. Referring to the participants, the requestor has to indicate two kinds of people, those whose presence to the meeting is mandatory and those who are only invited to participate. The agent contacts on behalf of the user, the personal agents of people involved in the meeting and tries to find a common agreement and to fix time and date of the rendezvous. Once the contract process is over the personal assistant notifies the requestor about the results of its activity, that is time and date for the meeting (for example Monday At Ten Thirty).

The negotiation protocol is an extension of the contract net [15] and envisages the presence of two kinds of roles the requestor - the initiator of the contract - and the participant - a person invited at the meeting. Personal Assistant agents perform these roles and drive the negotiation.

The negotiation protocol is described in figure 3.

The analysis of all the steps forming the process for arranging a meeting follows:

- *Step 0*: the requestor sends to participants a proposal for a meeting (date, place and time) and start waiting for responses.
- *Step 1*: the participant checks the personal calendar to verify if the time slot is empty or busy. If busy two situations are possible: the previously scheduled task has a lower importance and in this case the reply to the requestor is "I accept the proposal"; otherwise the participant rejects the proposal or offers to the requestor a new time and a new date for the meeting according to tasks already scheduled in the personal calendar.

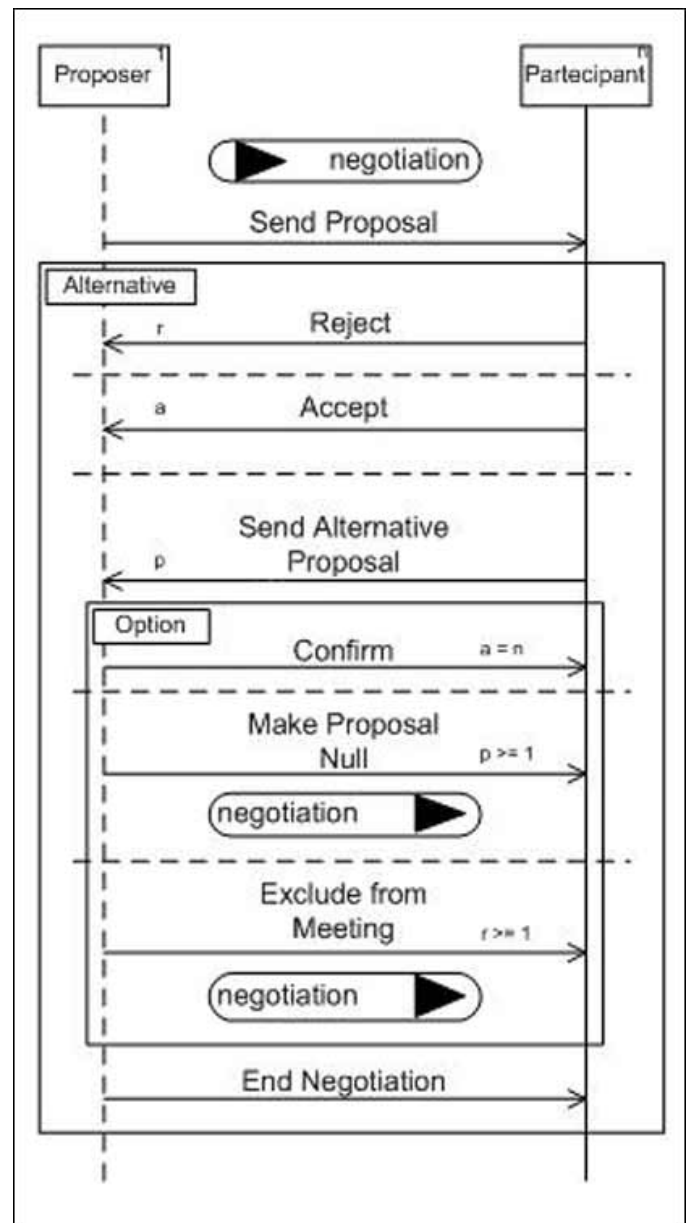


Figure 3. Negotiation Protocol

- *Step 2*: if the requestor receives a proposal agreement from all the participants the meeting is scheduled on date and time indicated in the proposal. If one of the compulsory participants rejects the proposal then the negotiation aborts. In the case a compulsory participant comes proposing an alternative date for the meeting the requestor checks the personal calendar and returns to step 0 with a new proposal according to date and time indicated by participants.

The algorithm defining the protocol is implemented in the Arranger behaviour of the Personal Assistant Agent and drives the negotiation accessing the Calendar knowledge. This negotiation protocol is a feasible and effective solution but it is possible to define new contract mechanisms for specific scenarios without changing the agent structure and the MAS architecture.

V. CONCLUSIONS

In this paper MATT a GCS application has been presented. MATT exploits the power of the agent-oriented approach and the flexibility of web services in order to obtain a system able to evolve according to the needs of the groupware users, easy to use and to manage, and accessible over the web from any type of client.

In particular the agent oriented paradigm has been a useful abstraction to model and to implement the system: agents are autonomous software entities able to expose a flexible behaviour. Agents acts on behalf of the users they represents, they can schedule meetings, sign-up appointments, and inform their owners of incoming events without the direct user intervention. Moreover, the ability to cooperate, compete, and negotiate with peers has been the key feature in implementing the negotiation protocol used to schedule meetings: agents solve the conflicts that arise when meetings overlap, and by looking at the hierarchical model described by MATT decide which event need to scheduled again. The use of a multiagent system is a winning solution in implementing a GCS application: a MAS defines the environment where the community of agents interact and provides them a set of services (i.e: DF, AMS, MTS) that make the implementation of many of the features required by a GCS straightforward.

The use of web services as a gateway technology to the entire system ensures that users can look up their agenda from any part of the world where there is an internet connection: users can use smart clients or just simple browsers to look-up their calendar and sign-up new appointments. Smart clients give user additional functionalities that make them able to work off-line and synchronize the data while connected. Again, the agent oriented approach has played a key role in the design of this functionality: a remote software agent living on the smart client takes care of the synchronization issues with the server, and registers all the appointments reserved by the user when he is working off-line. When the client connects to the system the agent interacts with its corresponding peer into the MAS and updates the calendar.

MATT is an interesting solution either by the use of the agent oriented technology that gives a powerful and high level abstraction to model the core functionalities of a GCS system, either by the use of the web service technology that ensures a wide, simple, and standardized access to the system.

REFERENCES

- [1] N.R. Adam, O. Dogramaci, A. Gangopadhyay, and Y. Yesha. *Electronic Commerce: Technical, Legal and Business Issues*. P T R PrenticeHall, Englewood Cliffs, NJ 07632, USA, 1999.
- [2] S. Vinoski, *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*, IEEE Communications Magazine, Vol. 14, No. 2, February, 1997.
- [3] N. Brown, C. Kindel, *Distributed Component Object Model Protocol -- DCOM/1.0*, available at <http://www.microsoft.com/oledev/olecom/draft-brown-dcom-v1-spec-01.txt>.
- [4] C. A. Ellis, S. J. Gibbs, & G. L. Rein, *Groupware: Some Issues and Experiences*, *Communications of the ACM*. 34(1), 38-58, 1991.
- [5] P. Wilson, *Computer Supported Cooperative Work (1st.ed)*. Oxford: Intellect, 1991.
- [6] L. Pallen, *Social, Individual, and Technological Issues for Groupware Calendar System*, 15-20 May, 1999.
- [7] R. B. White, *A Prototype for the Automated Office*. *Datamation*, pp. 83-90, April 1977.
- [8] M. Wooldridge, N. R. Jennings, "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*, 10(2):115-152, 1995.
- [9] S. Franklin, A. Graesser, "It is an Agent, or just a Program?: A Taxonomy of Autonomous Agents", *Proceedings of the Third International Workshop of Agent Theories, Architectures, and Languages*, Springer-Verlag, 1996.
- [10] G.Weiss, "Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence", G.Weiss Editor, MIT Press.
- [11] *Foundation of Intelligent Physical Agents (FIPA)*, <http://www.fipa.org>.
- [12] Vecchiola, C., Grosso, A., Gozzi, A., Boccalatte, A.: *AgentService*. *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'04)*. Banff AB, Canada (2004).
- [13] *Standard ISO/IEC 23271:2003: Common Language Infrastructure*, March 28, 2003, ISO.
- [14] C. Vecchiola, M. Coccoli, A. Boccalatte, "Agent Programming Extensions relying on a component oriented infrastructure", *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration (IRI - 2003)*, Oct. 26-29, 2003, Las Vegas, NV.
- [15] R. G. Smith, *The Contract Net Protocol: High-Level, Communication and Control in a Distributed Problem Solver*. *IEEE Trans. on Computers* C-29(12):1104-1113, 1980.