Ayllu: Agent-Inspired Cooperative Services for Human Interaction

Oskar Cantor, Leonardo Mancilla, Enrique González

Grupo de Investigación SIDRe, Pontificia Universidad Javeriana, Bogotá, Colombia

Abstract. Nowadays, people not only need to communicate, but also require to cooperate in order to achieve common goals. Thus, groupware applications are more complex, they have to satisfy new requirements of cooperation between people in many areas such as: organizations, industries or entertainment. Besides, due to the remarkable increment in the use of mobile devices and other technologies, groupware has found new environments to provide solutions and better services to end users. The Ayllu human cooperation model states that people must interact using well defined and structured protocols as rational cooperative agents do. This paper presents the Ayllu architecture, which is intended to develop groupware applications with a cooperative approach, called 5C paradigm; based on software agents that act as mediators between users working cooperatively. Cooperative services are constructed and executed by a mechanism called volatile group. Ayllu supports the execution of cooperative services, people can cooperate immersed in a pervasive environment, interact in an organized fashion, conform communities and pursue common objectives without concerning about their individual context.

1 Introduction

Most common people's work tasks are supported by communication. Nowadays, practically any application can be designed and implemented over different kind of devices, from a cell phone to complex distributed information systems. People can communicate without taking into account distance, time or place. However, new requirements emerge beyond simple communication applications; people needs to cooperate in order to achieve common goals. In fact, cooperation allows people to increase productivity, to reduce costs, to conform groups based on common interests, to negotiate, to facilitate decision making processes. For all these reasons, Groupware CSCW applications (Computer Supported Cooperative Work), arise as an alternative to fulfill these new requirements. CSCW applies within different contexts such as: industry, organizations, entertainment, among others. Taking into account the variety of devices that are available to support different applications, groupware can provide mechanisms for communications between people in real time without taking into account the place and respecting the individual context of each person.

Several models and tools oriented towards groupware have been developed that incorporate agent technologies [15], providing more flexibility and scalability. Some of the more developed frameworks of this type include the following ones: QuickStep

[1] MOTION [2], Proem [3], eNcentive [4], DISCIPLE [5], DACIA [6], DreamTeam [7], YCab [8], SALSA [9]. Each of these frameworks is based on a model that takes into account different groupware-oriented requirements. The use of peer to peer communication can be found in [2, 5, 3, 4]. Models based in N-tier architecture are used by [2, 4, 6, 7]. The concept of interaction protocols is present in [3, 7, 8, 9] and the notion of group services is employed in [2, 8, 9, 1]. Other interesting frameworks that propose different groupware requirements are: COOPSCAN [10], RCSM [11], that with [5, 7, 9, 4, 1], manage the group awareness abstraction. There also can be found an explicit groupware replicated architecture at [5, 6, 10, 14]; some cooperation models are introduced in [14, 15] and finally session management utilities can be found at [4, 5, 7, 10]. Most of these approaches are focused in providing architectural models and services to support human interaction; however, there are few works oriented towards the way people must cooperate.

The Ayllu human cooperation model states that people must interact using well defined and structured protocols as rational cooperative agents do. In fact, agents are social entities acting in a proactive way to achieve goals. Cooperative agents work together to fulfill collective goals, each agent does its best in spite of the multi-agent system objectives. In the groupware context, humans interact with each other in order to fulfill activities and accomplish objectives; based on this statement, in the Ayllu model, humans can be seen as rational agents that might behave in a similar cooperative way as agents do, and similar cooperation techniques, as those successfully used by agents, can be used by a human community. The Ayllu architecture includes mechanisms to facilitate the design, construction and use of human interaction protocols based on a cooperative agent approach. In this architecture, software agents act as mediators helping users to participate properly in cooperative actions. The framework is based on the abstraction of cooperative services, which include the definition of a schema where several persons participate in a systematic way to achieve a goal. The operation of a cooperative service is supported by the mechanism of volatile groups, which creates dynamically the required mediator agents. The model includes agents charged of the adaptation of the information to the user context and to different types of devices, also agents that aim to reduce user intervention.

In this paper, the Ayllu architecture will be introduced briefly: in particular, the important concepts of volatile groups and cooperative services. The focus of this paper is to explain the cooperative services creation process.

2 Ayllu Architecture Model

Ayllu consists of five basic layers and an access point mechanism, called CAP. Each layer is supported by the services provided on the lower ones. Figure 1 shows the general structure of Ayllu architecture. A more detailed description of Ayllu can be found in [17].

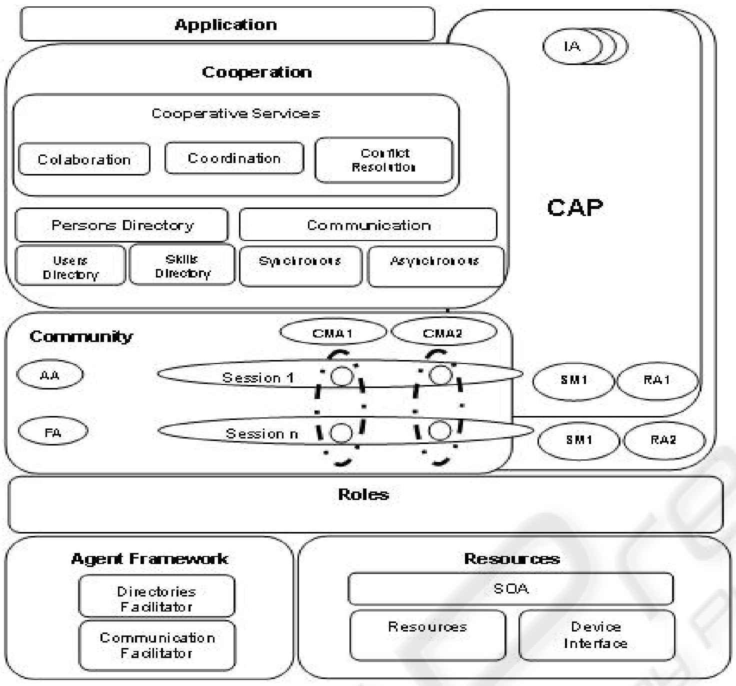

Fig. 1. The general architecture of Ayllu.

The five basic layers of the Ayllu architecture are the following:

1. Resources & Agent Framework Layer: is based on the MAD model [12], allowing to manage the limitations inherent of mobile devices. The basic agent framework that supports all higher layers is BESA [16], which provides the infrastructure to execute software agents.
2. Roles Layer: is an agent-oriented layer; it associates a set of responsibilities to generic types of agents, called roles. As new cooperative services are created, new roles should be included. Ayllu provides some basic roles, which can be used to create new generic cooperative services. These predefined roles include: the Community Manager Agent role (CMA) responsible for creating the community agents; the user Representative Agent role (RA) charged of representing the user in its interactions with the community; the Session Manager role (SM) acts as bridge between the user and community agents; the Interface Agent role (IA) acts as a flexible input/output mechanism; the Community Agent role (CA) is in charge of performing the cooperative tasks inside the volatile groups; the Administrator Agent role (AA) performs the management of the users and groups; the Factory Agent role (FA) responsible of the agent creation on demand.
3. Community Access Point: the Community Access Point (CAP), provides the access of users to the cooperative services supplied by the community layer. It has three basic elements: the Interface Agent (IA), the Representative Agent (RA) and the Session Manager Agent (SM). The CAP is also the one in charge of managing the user connection state.

4. Community Layer: the community layer contains the user's sessions and defines the active volatile groups. A session is the set of agents that allow a user to participate simultaneously in different cooperative services by means of the agents representing him in the active volatile groups.
5. Cooperation Layer: the cooperation layer is a user-oriented layer; it is the one in charge of providing the high level abstraction of the available cooperative services. The cooperative services offered by this layer provide the abstraction of collaboration, coordination and conflict resolution mechanism in the user context. This service abstraction offers an explicit way to access the cooperative interaction protocols mediated by the software agents that form a community. The programmer can design and easily include new cooperative services as required by the application requirements.
6. Application Layer: in the application layer resides any application that will make use of the provided cooperative services. Normally in this layer, the cooperative services are instantiated in terms of the application context.

3 Volatile Groups

A volatile group is a set of processes created dynamically with the purpose of carrying out a joint task; each one of these processes represents a user. These groups are created on demand and destroyed once the required task has been completed. In the Ayllu context, volatile groups are formed by agents, Community Agents and Community Manager Agents, and support the cooperative services provided by the groupware system. The volatile groups are the basis for the accomplishment of cooperative tasks in Ayllu. Hence, this mechanism allows the separation of the interactions between members of the cooperative community and the user interface access. As soon as a new cooperative service is requested by a user, a volatile group is created to deal with all the agent-mediated interactions associated to the service.

A volatile group is composed by a facilitator agent and a set of participant agents. The facilitator is the process that manages the group's life-cycle. The volatile group mechanism work in the following way (Figure 2):

1. The facilitator agent receives a service request generated by a user.
2. The facilitator agent identifies other users that could participate in the interactions derived from the requested service.
3. The facilitator agent creates new participant agents; each one represents a user that participates in the cooperative service.
4. While performing the service, the participant agents communicate to achieve a well-defined interaction protocol implementing a cooperation technique.
5. Once the task has been completed the volatile group is destroyed; in some cases, an answer can be returned to the user who made the initial request.

The volatile group mechanism can be used in a concurrent way. Several instances of a cooperative service can operate simultaneously. Furthermore, a user can participate in several cooperative tasks at the same time.

4 Cooperative Services

Ayllu proposes a new paradigm, the 5C paradigm, to manage people oriented groupware mediated by software agents. In this paradigm, people, accessing the system through different types of devices, are seen like agents that are part of a community, performing their cooperative tasks by means of software agents (Figure 2). This approach gives as result an architecture for people oriented groupware in which software agents are capable of making their own decisions on behalf of the user without permanent human interaction. A detailed explanation of the 5C paradigm is beyond the scope of this paper; more information about it can be found in [17].

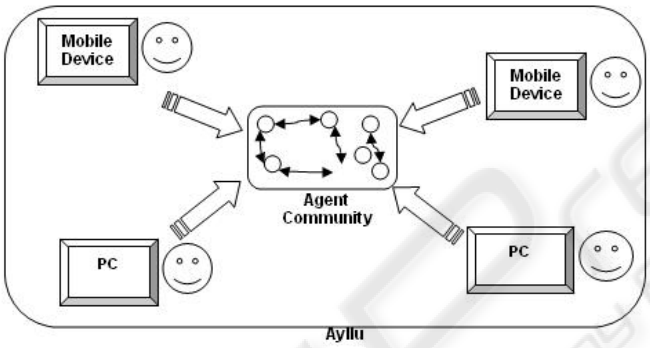

Fig. 2. Ayllu's general approach.

A cooperative application is implemented by a set of *cooperative services*. A cooperative service facilitates the execution of a well-defined cooperative task between concerned users. Software agents are dynamically created to perform all the necessary tasks to achieve the purpose of the service, respecting an interaction protocol that defines the way intentional messages with a clear semantic are exchanged between the adequate persons. In Ayllu, this group of agents conform a volatile group, as explained earlier. Each concerned user communicates with its community agents through its CAP; in particular, the session manager SM act as a bridge between the agent community and the user's CAP.

As figure 3. illustrates, any cooperative service in Ayllu requires three basic agent types in order to work: a *Community Manager Agent (CMA)*, some *Community Agenst (CA)* and a *Factory Agent (FA)*. The FA is the agent in charge of creating the CMA agent required to start a cooperative service. There's only one FA for an entire application and the programmer must define which services will provide the application and the types of CMAs it creates. Once the CMA of a requested service has been created, it perform a look up on the *person's directory* to find other users that may participate in the service execution; the CMA creates the volatile group of CA agents, each one representing one user. These three types of agents must be specialized by the programmer for each specific cooperative service.

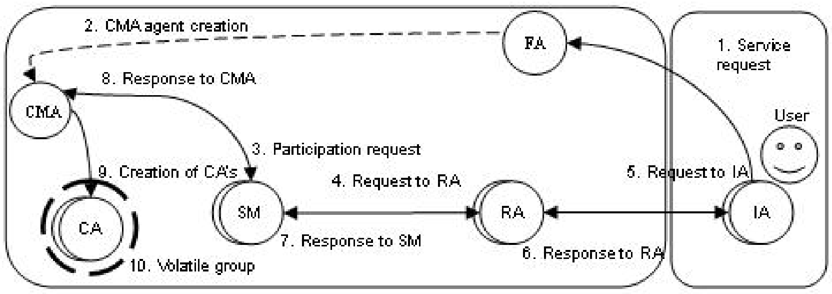

Fig. 3. Cooperative Service creation process in Ayllu.

In order to create a cooperative service in Ayllu, six steps must be followed:

1. Identify the cooperative services that the application will provide.
2. Identify the human agents (users) that will take part in the service.
3. Identify the software agents required to perform the task and the different roles that they might take.
4. Create an interaction diagram, including software agents and users.
5. Establish the generic classes for the service in order to define the responsibilities for each agent (role definition). These generic classes can be reused if needed.
6. Specialize the generic classes to obtain a cooperative service to cope with an application's specific requirements.

These steps will be explained in detail in the next section. The basic distribution of the Ayllu framework provides a series of cooperative services which can be used by any groupware application. These services are based on the abstract model previously explained. These pre-defined services include the following functionalities: Group of Interest and Presence Detection; Distributed Artifact Search, for managing shared resources; Chat Service provides synchronous communication; Contract Net, for task allocation, it is a task allocation technique inherited from cooperative agents; Messaging Service provides asynchronous communication.

The programmer can create new cooperative services to cope specific application requirements. Remark that the *Contract Net* service is a good example of how multi-agent system cooperation techniques can be adapted to be used in groupware cooperative scenarios, many other agent cooperation techniques can be implemented.

5 Creation of Cooperative Services in Ayllu

As mentioned earlier, Ayllu allows the creation of multiple generic types of cooperative services that can be reused in different situations. These generic services can be specialized according to the requirements of any cooperative application.

5.1 Creation of Generic Cooperative Services in Ayllu

To create a generic cooperative service in Ayllu, a set of well defined steps must be followed, as mentioned in section 4. If these steps are followed in the appropriate way, the development of cooperative services will be more efficient and the obtained results will also be more reliable. In this section, these steps will be explained using the *Contract Net* service as an example.

1. Establish the requirements of the application in order to identify the required cooperative services. For the *Contract Net* service, the main requirement is to provide a mechanism for users to make mutual agreements for task allocation.
2. Identify the types of users that will take part in the service. Users have different needs of information and interaction with other users, depending on the activities they perform and the roles that each one of them has in a given context. In a *Contract Net* situation, there are two types of users: the *manager* and the *contractor*. The manager sends offers to the contractors, which can decide to respond with a proposal; the manager analyzes the proposals and decides which contractor must perform the task. The contractors are informed of the decision.
3. Identify the software agents required to perform the task. As mentioned earlier, Ayllu's cooperative services are supported on three basic agent types.
4. Create an interaction diagram to specify the relationships between the community agents. This diagram shows all the events that the agents must handle, and will be used later in the implementation phase. Figure 4 shows the interaction diagram for the *Contract Net* service. As can be seen, once the CA has been created, the CMA sends a task request to the CA manager. This agent sends the task request to the CA contractors, which ask their users for a bid. Once the bids have been collected, the CA manager selects the best ones and sends them to its user, who decides which one fits better his needs; finally, the CA manager informs the contractors about the decision.

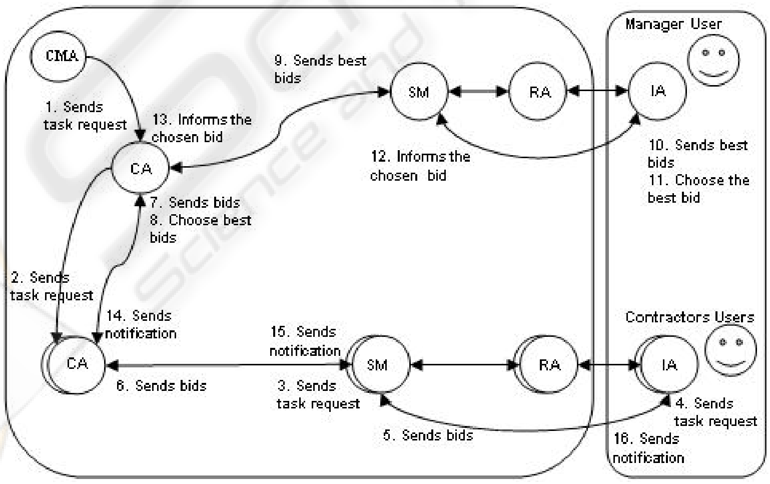

Fig. 4. *Contract Net* service interaction diagram.

Notice that interaction diagrams are not intended only to describe a sequence of activities; the main goal of interaction diagrams is to provide a mechanism to identify and characterize interactions between the agents used as mediators.

5. Define the generic classes for the cooperative service, which implies to extend from the generic classes provided by Ayllu. These service role classes implement the functionality and protocol of the general agents specified in the previous steps. The *Contract Net* service requires the CMACnet specialized community manager agent and CAContractNet community agent. The second one can be built to take one of the two roles described earlier for the service, manager or contractor. For the *Factory Agent*, only the list of services that the application will provide must be defined, in this case the *Contract Net* service.
6. Finally, specialize the generic service taking into account the real application context. This procedure implies to extend the generic service role classes. This extension normally does not include any protocol change, only the methods already invoked by the service roles must be supplied. For instance, if the *Contract Net* service is used to negotiate a medical task allocation in a telemedicine application, the specialized agents should have the necessary knowledge to handle the relevant information, i.e. the medical staff information or the patient's medical history.

5.2 Creation of Specialized Cooperative Services

In order to illustrate the creation of a specialized cooperative service in Ayllu, a telemedicine application example will be used. The first step is to define which services will be provided by the application. An entry in the *Persons Directory* must exist for each service, describing and associating it with the users that may use it during the execution of the application. A new *Factory Agent* must be created specifically for the purpose of creating the CMA agents according to the service requirements. The *Factory Agent* also verifies the permissions of the user who invokes the service. The general structure of the generic *Contract Net* service in Ayllu is shown in figure 5.

The CMAContractNet class provides the functionality of a community manager agent and also controls the volatile group life cycle. It also associates responsibilities to the service participants: it gives the responsibilities of a contract net manager to the user who started the service, the other users involved are granted with the contractor responsibilities. The CAContractNet is the agent that can assume any of the two roles defined for the generic service, as mentioned before. The CAContractNet and the CMAContractNet do not need to be specialized if a basic *Contract Net* service is required; if any modifications to the interaction protocol need to be made, inheritance will be required in order to do so.

The ContracNet class must be implemented by inheritance to suite the specific requirements of information. In this class, the attributes to handle the *Contract Net* service interactions should be defined by the programmer.

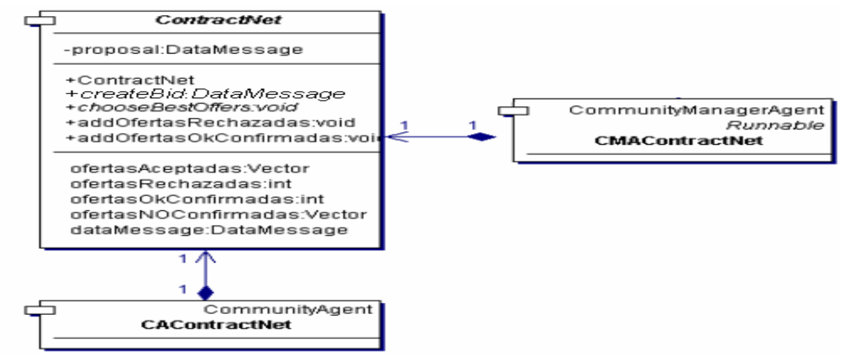

Fig. 5. Generic *Contract Net* service structure.

In the *Contract Net* service, the methods that need to be implemented are `createBid` and `chooseBestBids`. They include the application logic to construct a proposal in response to a task request, and to evaluate the collected bids. After completing all the specialization process, the cooperative service is ready to use by the cooperative application designed using the Ayllu framework.

As this section shows, Ayllu provides a simple way of creating cooperative services, reducing development times and making a contribution to the CSCW field.

6 Conclusions

The increased development of mobile computation opens the possibility to design and implement applications that will be capable to adapt and to satisfy the present information needs that technology users require. In particular, user mobility facilities must be provided in order to integrate people and services in a flexible way. The groupware paradigm has been developed during years, oriented to the traditional desktop devices like PCs or laptops; however, taking advantage of the possibilities offered by the new wireless technologies requires not only a technical approach, but also new paradigms centered in buster people cooperation.
The Ayllu´s human cooperation paradigm, contemplates the use of software agents to support people cooperation; the paradigm states that users interact to collaborate assuming tasks, coordinate planned actions, and solve conflict using MAS inspired techniques; people is seen like being part of a multi-agent system, they act as human rational cooperative agents. Ayllu is not only a groupware oriented architecture mediated by agents, but also a paradigm for groupware development. The concept of cooperation between people is extended through the implementation of software agents that serve as mediators for their activities, facilitating the accomplishment of a common objective in a successful way. The basic services provided by the Ayllu framework reduce the development time of a cooperative application. Actual work includes the creation and implementation of new generic cooperative services inspired from SMA techniques, and also the identification of specialized cooperative services for a business organization environment.

References

1. Roth, J., Unger, C.: Using handheld devices in synchronous collaborative scenarios. Department of Computer Science-University of Hagen (2001).
2. Kirda, E., Fenkam, P., Reif, G., Gall, H.: A service architecture for mobile teamwork. Technical University of Vienna (2002).
3. Kortuem, G.: Proem: A middleware platform for peer-to-peer computing. Mobile Computing and Communications Review, Volume 6, Number 4, (2002).
4. Ratsimor, O., Finin, T., Joshi, A., Yesha, Y.: eNcentive: A framework for intelligent marketing in mobile peer-to-peer environments. Departament of Computer Science and Electrical Engineering, University of Maryland Baltimore Country (2003).
5. Wang, W., Dorohonceau, B., Marsic, I.: Design of the disciple synchronous collaboration framework. The State University of New Jersey (1999).
6. Litiu, R., Prakash, A.: Developing adaptive groupware applications using a mobile component framework. Department of Electrical Engineering and Computer Science, University of Michigan (2000).
7. Roth, R., Unger, C.: Dreamteam a platform for synchronous collaborative applications. Department of Computer Science- University of Hagen (2000).
8. Buszko, D., Lee, W.-H., Helal, A.: Decentralized ad-hoc groupware api and framework for mobile collaboration. Motorola iDEN Group, CISE departmen, University of Florida (2001).
9. Rodriguez, M., Favela, M.: A framework for supporting autonomous agents in ubiquitous computing environments. Departamento de Ciencias de la Computación, CICESE, Ensenada, Mexico (2003).
10. Balter, R., Atallah, S. B., Kanawati, R.: Architecture for synchronous groupware application development. HCI, Tokyo, Japan (1995).
11. Yau, S. S., Karim, F., Wang, Y., Wang, B., Gupta, S. K.: Reconfigurable context sensitive midleware for pervasive computing. Computer science and Engineering Department, University of Arizona (2002).
12. Botero, A., Giraldo, H., Moyano, A.: Mad: Arquitectura basada en componentes distribuidos para dispositivos móviles. Proyecto de Grado para optar por el título de Ingeniero de Sistemas, Pontificia Universidad Javeriana, Noviembre (2004).
13. Roth, J.: A taxonomy for synchronous groupware architectures. Department of Computer Science - University of Hagen (2000).
14. Gerosa, M. A.: Towards an engineering approach for groupware development. Software Engineering Laboratory, Computer Science Department, Catholic University of Rio de Janerio (2003).
15. J. Ferber, Multiagent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.
16. González, E., Bustacara, C., Avila, J.: Besa: Behavior- oriented, event-driven social-based agent framework.". PDPTA'03, Las Vegas-USA, CSREA Press, vol. 3, Junio 2003, pp 1033-1039.
17. Cantor, O., Mancilla, L., González, E: Ayllu: A Cooperative Approach to Groupware Development. In proceedings of IEEE-ICC International Conference on Communications (2006).