An Architecture for Modeling Internet-based Collaborative Agent Systems

Roberto A. Flores[1], Rob C. Kremer[1], Douglas H. Norrie[2]

[1] Department of Computer Science, University of Calgary, 2500 University Dr. NW, Calgary, Canada, T2N 1N4
{robertof, kremer}@cpsc.ucalgary.ca
[2] Department of Mechanical & Manufacturing Engineering, University of Calgary, 2500 University Dr. NW, Calgary, Canada, T2N 1N4
norrie@enme.ucalgary.ca

Abstract. This paper describes an architecture for modeling cooperating systems of communicating agents. The authors' goal is not that of providing a framework to implement multi-agents systems (there are tools—such as CORBA, Java and DCOM—that do an excellent job on that), but rather to provide an architectural metaphor upon which collaborative multi-agent systems could be modeled. The approach is based on requirements defined with a practical view of the communicational and resource-oriented nature of distributed collaborative multi-agent systems.

1 Introduction

Increasingly, the Internet will be used for commerce, industry, and educational interactions between multiple parties. These interactions are intermittent but sustained over days, months, and years. They will involve multiple sources of information and often record, transform, and store considerable quantities of information for subsequent access and re-use. The Collaborative Agent System Architecture (CASA) presented in this paper provides a structural framework to support the modeling of such distributed, collaborative multi-agent systems.

In this architecture, agents are seen as software entities that pursue their objectives while taking into account the resources and skills available to them, and based on their representations of their environment and on the communications they receive [1]. In the case of agents in collaborative systems, agents are also capable of delegating task realization to agents capable of performing the required task. Internet-based systems pose an additional challenge to this scenario in that agents collaborate in a dynamic, distributed environment where agents from heterogeneous sources could interact.

It is common to find collaborative mechanisms in currently available implementation frameworks; however, modeling systems based on these facilities result in systems with less flexibility and adaptation to changes given previous design commitments to the underlying framework. The collaborative architecture presented here aims to separate the modeling of multi-agent systems from the specifications that

designers need to commit given the low-level mechanisms of proprietary frameworks used in the implementation of multi-agent systems.

There are three elementary components that we have identified as fundamental in the design of collaborative multi-agent systems: computer resources, agents, and owners. These are interrelated, since both agents and computer resources are bound by the regulations set by the human institutions owning and controlling those agents and resources. The latter are further described below.

- Computer resources: Computer resources are hardware and software resources that are available to agents for the execution of their tasks
- Agents: Agents are communicational and collaborative entities that perform their duties using the computational resources available to them. Agents can take the role of requesters or suppliers without these being one of their intrinsic characteristics. Agent requesters and providers rely on the definition of roles and on mechanisms of advertisement to locate other agents that can perform tasks for them.

2 ARCHITECTURE REQUIREMENTS

In this architecture, we model communities of agents in two ways: based on the computer resources agents use, and on the communicational contexts upon which they interact.

On the one hand, agents are seen as communicational entities that interact with other agents to achieve their goals. In this view, agent communities are formed as a result of interactions and the preferred locations in which interactions take place.

On the other hand, agents are seen as entities with affiliation, performing their tasks on behalf of human institutions that endorse and exert authority over them. In this view, affiliation enables resource-oriented communities by binding agents and resources. It is common that affiliation is addressed in terms of the low-level mechanisms implemented by different frameworks. We believe that these decisions should be reflected at the modeling stage, independently from the proprietary mechanisms offered by implementation frameworks.

Based on the concepts introduced above, we present several minimal requirements upon which we base our architecture.

1. The architecture should provide means (for humans) to organize computer resources in identifiable clusters.
2. The architecture should provide means to control agents' usage of computer resources according to human policies.
3. The architecture should provide means for agents to locate other agents for the purpose of collaboration and delegation of tasks.
4. The architecture should provide means for agents to create and maintain settings where agents interact (settings may include the state and history of agent interactions, and any artifacts pertinent to the context of interaction).

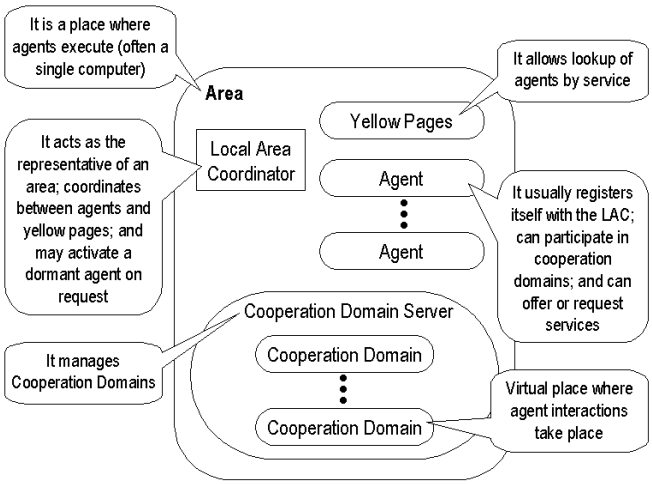

Fig. 1. Basic concepts in the CASA architecture

3 The Collaborative Agent Systems Architecture

In this section we describe a structural framework for the modeling of collaborative agents in distributed environments based in the requirements from the section 2.

Basic to this architecture are the concepts of areas (bounded resource-oriented regions), each of which contain a local area coordinator (an agent that is—in principle—responsible to negotiate with other agents the use of resources in an area), yellow pages (agents to which services offered by agents are advertised and queried), and cooperation domains (virtual contexts of interaction that may last over time, and which are supported by cooperation domain servers). These concepts (illustrated in Figure 1) are further described in the subsections below.

3.1 Areas

To satisfy the requirement that "The architecture should provide means (for humans) to organize computer resources in identifiable clusters" (requirement 1) we include the concept of areas.

Areas are bounded regions comprising computer resources owned by human institutions. Areas could be formed of partial resources from one computer, an entire computer, or a group of computers. Agents in areas form communities whose common denominator is being accepted under the usage policies and regulations set by the owners of the area's resources. Just as in a human club or sports association, membership is granted based on regulations that members are expected to abide by. In this view, membership does not necessarily imply awareness of other members of the area,

since areas are facilities-oriented associations rather than associations for communication and interaction among the members. This is not to say that the latter is not important: cooperation domains are explained a later sub-section.

3.2 Local Area Coordinators

To satisfy the requirement that "The architecture should provide means to control agents' usage of computer resources according to human policies" (requirement 2) we include the concept of local area coordinators.

Local area coordinators (LAC) are agents who put into effect the usage policies given by the human institutions owning the computer resources allocated in an area. Agents requesting the use of resources in an area need to go through a registration process with the LAC in that area. Agents succeeding in their registration are expected to surrender some of their autonomy in exchange for the use of resources. Depending on the usage policies assigned to the area, agents' executable code could be required to reside in one or more of the area's resources, and its use of resources could be influenced by the usage regulations assigned to these resources.

In addition to provide an abstraction for a group of computer resources, areas and LAC can also help to encapsulate the execution state of registered agents. For example, if a communication is requested for a non-running agent or an agent whose execution is suspended, the LAC could allocate resources to enable that agent to resume execution and react to the communication, or it could refuse the communication on the basis of the agent unavailability.

3.3 Yellow Pages

To satisfy the requirement that "The architecture should provide means for agents to locate other agents for the purpose of collaboration and delegation of tasks" (requirement 3) we include the concept of yellow pages.

An agent attempting a task that requires more resources or abilities than it is capable of supplying could break down the task into several sub-tasks, and then distributed these among several agents. To succeed, this division of labor should take into account the functionality provided by agents and the actions required by the tasks that are delegated. This understanding of an agent's functionality (i.e., what the agent can do) can be based on roles (skills that are known to be performed by individuals supporting that role), or on the identities of specific instances (skills that specific agents are known to perform). From these, roles have the advantage that they allow agents to plan a division of labor without committing to any specific agent instance. This is of importance given that agents can unpredictably appear and disappear from the environment.

In any event, agents that offer or request services need to rely on mechanisms for the advertising and requesting of services. Agents can make available a set of skills (represented as a role) for others to request, or they could post the need for an agent with a specified set of skills. These two functions are akin to that of yellow pages and

job posting boards, respectively. In the case of the CASA architecture, we define an agent called Yellow Pages to support these functions. Yellow Pages are just a special case of ordinary agents. An agent may access yellow pages through a database of known yellow pages kept by LAC.

3.4 Cooperation Domains

To satisfy the requirement that "The architecture should provide means for agents to create and maintain settings where agents interact" (requirement 4) we include the concept of cooperation domains.

Cooperation domains can be conceptualized as virtual places in which purely communicational resources (i.e., resources that are manipulated through communications) are made available to agents in the place. The functionality of cooperation domains can be paralleled to that of Internet multi-user dungeons, where participants enter rooms in which objects are located and shared, and where agents' actions can be perceived by all other participants.

Our rationale in pursuing this functionality is based on the fact that meaningful interactions in dynamic environments cannot be accomplished on the sole basis of message exchange [3]. In this view, we devise cooperation domains as entities where messages are linked to the settings in which they are produced. In the architecture, cooperation domains are supported and maintained by cooperation domain servers.

Possible applications for cooperation domains include that of acting as centralized state and message repositories. One example is the one implemented in jKSImapper [2], where agents join cooperation domains to access and modify (using messages) the state of a diagrammatic structure shared among the participants of the cooperation domain.

4 Architecture Implementation

This section addresses the implementation of conversation policies for the agents we have built for the architecture. The current implementation was done using the Java programming language; and messages are communicated in KQML format.

In the CASA architecture, agents are expected to recognize and exchange messages by following agreed-upon conversation policies [3] in the domain of interaction.

Conversation policies are patterns of communication to be adhered to by interacting agents. These policies define causal-relation sequences of messages communicated among types of agents; and they describe how agents should react to these messages during interactions.

We have defined several basic policies that enable agents to interact in the architecture. These are:

- Registration: To register (to a LAC) as part of an area.
- YP Locations: To query a LAC about known yellow pages.
- Advertisement: To advertise services in Yellow Pages.

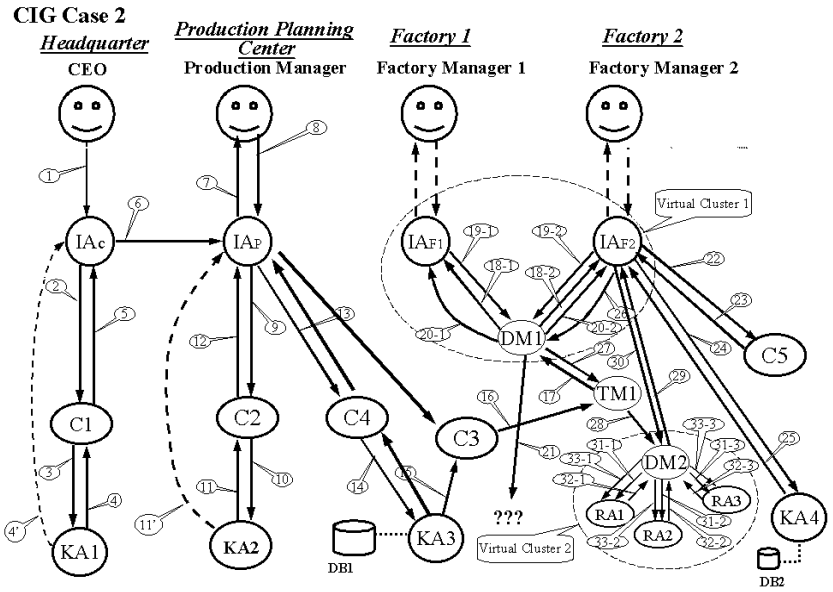

Fig. 2. A production planning scenario

- Search: To query advertised services in Yellow Pages.
- Cooperation Domain Subscription: To join a Cooperation Domain.
- Cooperation Domain Invitation: To invite another agent to join a CD.

In addition to these primitive conversations, it is expected that particular domain-specific applications will develop and include new conversation policies appropriate to the agent interactions in their domain.

5 Modeling a Production Planning Multi-Agent System

Figure 2 shows a multi-agent system scenario in the domain of production planning. This scenario was modeled using the Infrastructure for Collaborative Agent Systems [4], which is a high-level infrastructure composed of both vertical and horizontal functional layers.

5.1 Test Case: Brief Overview

The production planning test case scenario (illustrated in Figure 3) is composed of a number of agents, including humans. These agents are, at the top level, humans that interact with interface agents (IA), which then interact with collaboration (C) and mediator (DM and PM) agents, some of which interact with knowledge management (KM) agents, found at the bottom level in the figure.

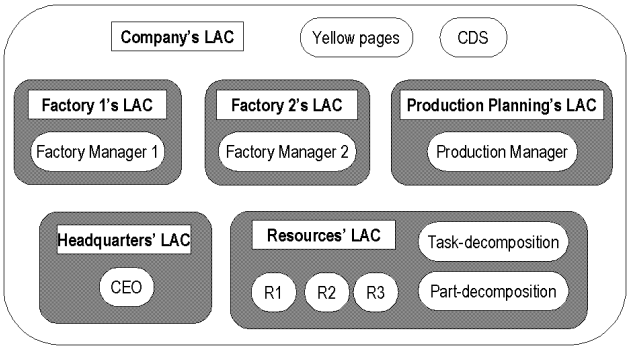

Fig. 3. . A CASA structural model for the Production Planning scenario

The dynamics of this scenario are as follows (a more detailed description is found in [5]): Initially, the CEO receives a production order for a product B, and based on this request it looks for a production manager to carry on with the order. This process is realized as follows: the CEO asks her interface agent (IAc) for a suitable production manager. IAc asks a collaboration agent (C1) where to find an agent knowledgeable in production managers, leading (in this case) to KA1, which then provides the requested information back to IAc, etc.

5.2 Modeling the Test Case using CASA

There are several ways on which this scenario could be modeled using the CASA architecture. For simplicity, we assumed that all agents in the scenario belong to the same company. Figure 5 shows one possible model. In this case, one company-wide area is encompassing all agents. Under this approach, it is expected that general organizational regulations could be applied by the LAC at this level. Next, we defined a yellow pages and cooperation domain server to assist in the interaction of agents within the company.

The interaction dynamics of the test case scenario under this model are as follows (for simplicity, we assume that all agents have registered with their respectively areas and that all service provider agents have advertised to the yellow pages—the dynamics of these interactions should be observable from the descriptions below).

Initially, the CEO receives an order for product B, and sets off to find about available production managers for this order. In this example, the (human) CEO and the IAc are considered under the single agent CEO. The agent CEO contacts the headquarters LAC and asks about known yellow pages. The headquarters' LAC then forwards the request to the company's LAC, and a reply with the location of the YP is sent back to CEO. The CEO then queries the YP for known production managers, to which the yellow pages replies with the location of the production manager in the production planning area. The CEO sends a request to the production manager for the production of the order (this, through a application specific conversation policy), etc.

6 Summary

The Cooperative Agent Systems Architecture is a software model that aims to simplify the implementation of multi-agent systems in a flexible and minimally intrusive way. The various architectural components are derived from four fundamental specifications, and further implied implementation details, such as conversation policies, and a possible agent model are also described. Finally, the architecture is applied to an example problem taken form the production planning domain.

One of the principal advantages of this model is that it supports agent discovery with a minimum of meta-information to be provided to agents. Specifically, the only information that agents need to be given is that of the location of the LAC where they initially subscribe as part of a system. Once agents are able to register to the specified area, they will have the means to locate agents that have advertised services.

Acknowledgments

The authors thank all other members of the Intelligent Systems Group at the University of Calgary for their contribution to the development of the Collaborative Agent System Architecture, and detailed discussions on the case study. Financial support for this work is provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and SMART Technologies Inc. of Calgary.

References

1. Ferber, J. (1999) Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison Wesley Longman.
2. Flores, R.A. (1997) Programming Distributed Collaboration Interaction Through the WWW. M.Sc. Thesis, Department of Computer Science, University of Calgary, Canada, June, 1997
3. Greaves, M., Holmback, H., and Bradshaw, J. (1999) What is a conversation policy? Proceedings of the Third International Conference on Autonomous Agents (Agents 99), Workshop on Specifying and Implementing Conversation Policies, M. Greaves and J. Bradshaw (eds.), pp.1-9, Seattle, WA.
4. Shen, W., Norrie, D.H. and Kremer, R.C. (1999) Towards an Infrastructure for Internet Enabled Collaborative Agent Systems. Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99), B.R. Gaines, R.C. Kremer and M. Musen (eds.), Vol. 1, pp. 3-3-1: 3-3-15, Banff, Canada.
5. Shen, W., Norrie, D.H., and Kremer, R., (1999) Developing Intelligent Manufacturing Systems Using Collaborative Agents. Proceedings of the Second International Workshop on Intelligent Manufacturing Systems (IMS 99), pp. 157-166, Leuven, Belgium.