Multi-agent System Architecture for Collaborative E-Commerce

Xiaojun Shen, Shervin Shirmohammadi, Chris Desmarais and Nicolas D. Georganas

Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER)
School of Information Technology and Engineering (SITE)
University of Ottawa, Canada
email { shen | shervin | chrisd | georganas}@discover.uottawa.ca

Abstract

In this paper, we present the design and implementation of and interdisciplinary research project involving an intelligent agent-based framework for collaborative e-commerce applications. A Multi-Agent System (MAS) architecture for large collaborative e-commerce environments is designed and developed, where a number of geographically dispersed users (customers/merchants) can participate.

1. Introduction

Existing e-commerce applications provide users with a relatively simple, browser-based interface to access available products and services. These applications often lack in the emulation of the social factor. The customers are mainly kept separated and everyone is shopping, as if s/he was in an empty shop. Thus, customers are not provided with the same shopping experience, as they would be in an actual store or mall. Shopping is a social activity people enjoy doing along with friends and relatives. In particular, it is likely that shopping is an activity that is socially facilitated, meaning that when shopping in the company of others, people engage in it more often and enjoy it more. Marathe [1] states “people don’t like to shop in an empty store.” To substantiate this opinion, he cites a survey, which shows that 90% of shoppers prefer to communicate with others while shopping. Warms et al [2] argue for shopping communities because they “increase stickiness (customer loyalty) [and] viral marketing (word of mouth), reduce the cost of customer acquisition, and drive higher transaction levels.” Considering the current growth of e-commerce on the Web and the desire to make shopping as easy, natural and enjoyable as possible, it would be interesting to enhance the way people currently shop on the Web by adding support for more collaboration between customers and salespersons or among customers. Therefore, providing an e-community web shopping experience makes on-line shopping closer to the actual experience people have in real shopping environments. The industry has also acknowledged this concept, and is now seriously looking at collaborative e-commerce. For example, WebSphere Commerce Business Edition from IBM provides features of real-time collaboration among a group of buyers or sellers such that they could share documents, discuss a contract and negotiate terms in a private electronic workspace.

One of the advantages of applying e-communities in e-commerce applications is the enhanced interactivity between merchants and buyers, and between customers and visitors. It enables online merchants to offer features that are lacking in most of today's e-commerce stores. For example, the community online shopping mall makes it easy for storeowners to provide real-time customer support, sales assistance, cross-selling, promotion and individualized care that have traditionally been proven to improve sales [3].

The purpose of this interdisciplinary research is to design an intelligent agent-based framework for collaborative e-commerce applications. We aim to develop Multi-Agent System (MAS) architecture for large collaborative e-commerce environments where a number of geographically dispersed users (customers/merchants) can participate. Collaborative commerce is realized by the interactions among agents in the e-commerce community. The rest of this paper is organized as follows. In section II, a multi-agent system for collaborative commerce implemented over Microsoft .NET framework is proposed. Section III depicts the design and implementation of e-commerce communities over the proposed MAS system. Finally, the summary of the presented research is described in the conclusion.

2. Multi-Agent System for Collaborative Commerce

In order to maximize adaptability and flexibility in an e-commerce environment, this paper proposes an architecture for creating e-communities as a collection of related agents - each agent responsible for a specific task. By working together, the group of agents is able to solve more complex system demands. By breaking a large e-commerce system into sub-tasks, the entire system becomes more encapsulated and adaptable. The ability to solve complex requirements emerges from the interoperation of different agents and potentially the interoperation of different agent communities.

A. *Generic Architecture for Agent-Based Collaborative Commerce*

In our previous work, the AGILE architecture was proposed [4]. This is an architecture for agent-based collaborative and interactive environments. This research expands on the previous work. The proposed system architecture is shown in Figure 1. It is divided into two closely coupled logical modules: the *information exchange* and the *coordination* among the system components and the agents, and the design and cooperation of the agents themselves. These agents are used to interact with the user, offer a homogeneous interface, and support collaborative work between different users. The *Agent Cluster*, a surrogate of a user in the distributed system, consists of a number of agents (user agent, shopping agent, sales agent, etc.) which provide the user with a homogeneous interface for various activities. They also trace the user behaviors to learn about the user's preferences, to communicate with other users, and to perform tasks for the user even after s/he has logged out. The *Directory* provides distributed white and yellow page services to deliver static information about the locations and addresses of agents and information databases, which are distributed on the network. The *Software Bus*, which is designed based on Microsoft .NET framework, is responsible for inter-agent communications.

B. *An Agent*

In our context, an agent is a software component running in distributed environments and capable of performing independent actions to process requests from other agents, or from external applications. The handling of these requests will often require making new requests of other agents in the system. An agent in the system has three required elements (Figure 2): an ***address***, a ***logic component***, and a ***published interface***. Almost all agents will also have a name property.

Address

The address property is used to locate the agent in the distributed environment. The proposed system in this paper is implemented over Microsoft .Net framework, and in that environment the address is an http address (ie. http://demomachine:5050/demoAgent).

Logic Component

The logic component is fairly open. Behind the agent interface there needs to be an application that will handle the request. Whether an old legacy system, or entirely new code, there is something behind the interface that handles the request and creates reasonably intelligent responses to requests. There is no hard requirement as to how this is done; it may be as simple as a database lookup or calculation, or it may require the use of complex machine learning algorithms. The logic required for a specific agent is dictated by the needs of that agent, and the types of requests it is expected to handle.

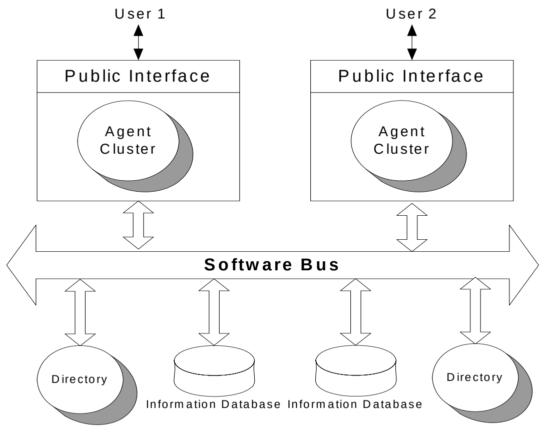

Figure 1 Generic Architecture of Agent-Based Collaborative Commerce

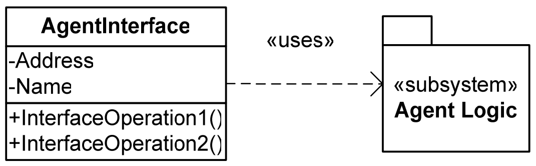

Figure 2 Agent Overview

Interface

The interface property is what allows other agents or external applications to communicate with and access the agent. The approach is to use standardized generic interfaces. Typically, this involves writing an interface structure that will be used by several different types of agents. Communication among agents is achieved through an agent communication language: the Knowledge Query Manipulation Language (KQML) [5].

C. *Agent-Based Community*

A community, in the proposed architecture, is a group of related agents (Figure 3). Agents in a community are realized using the interfaces required by that specific community, and expect other agents in that community to understand the known interfaces. The agents in a community are also expected to share *Naming Service Agents*, so that agents (and applications) can find other agents in the community. By grouping agents inside communities, other agents and applications are able to find and make use of the agents in that community. There are a few other types of agents in a typical community.

Naming Service Agent

The Naming Service Agent is a special purpose agent that exists to maintain system knowledge of the existence of agents in a community. The naming service is responsible for maintaining its own knowledge about the agents in a community (typically by simply servicing add/remove agent requests that are sent from other agents when they enter or leave the system). It then shares this knowledge when an authorized agent or application needs to find an agent.

The reason the Naming Service Agent is an integral part of a community is that it is the only agent that will always be known by its address. Agents are typically transient, able to move, enter or leave a community based on the specific tasks of the agent. Because the Naming Service Agent provides access to other agents in a community, Naming Service Agents actually define what agents exist in a specific community and the boundaries of what exists within its community.

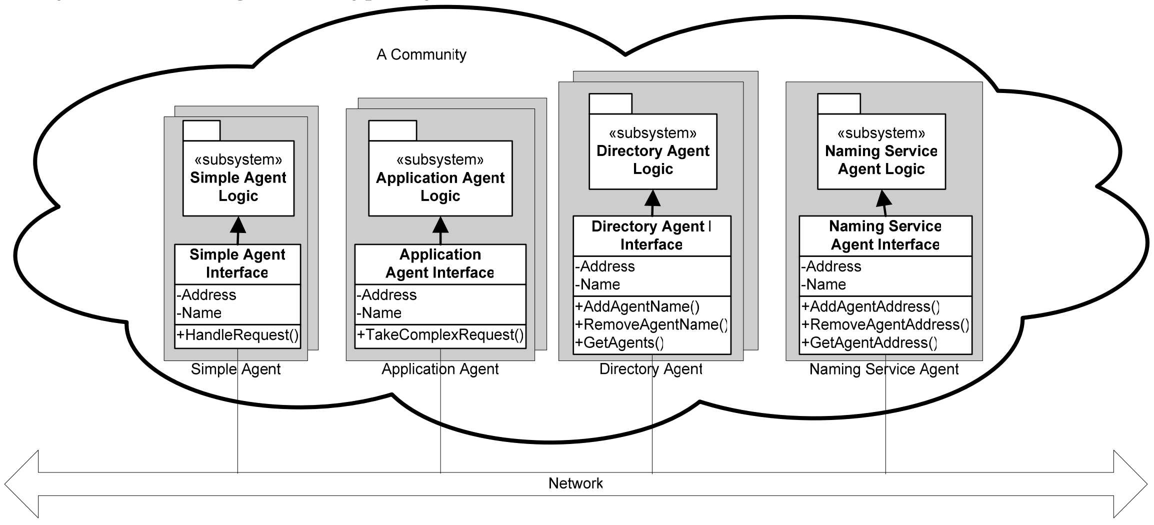

Figure 3 Agent-Based Community

In many cases, a simple address lookup will be insufficient for community needs. When security or privacy control is required by a community a ticket generating server will act as a naming service. Commonly, naming services and ticket generators allow agents to find and contact resources in a community. The exact mechanics differ according to community needs.

Directory Agent

Directory Agents provide known lists of agents that have registered to perform a specific task. All agents capable of taking orders might register with a single agent that keeps a list of “order taking” agents. This is similar to the job done by the Naming Service Agent, but all agents in a community should register with the Naming Service Agent and only agents that want specific requests should register with a Directory Agent. Directory agents usually have interface methods for adding and removing agents.

Simple Agent

Simple Agents are agents that perform a very specific task of processing requests without maintaining data about the other agents in the system. They are aware of the Naming Service agent because they will usually register when they enter or leave a system. They may also be aware of Directory Agents for similar reasons. A Simple Agent is simple because it can process some requests without relying on other agents. Simple agents require methods directly related to their purpose

Application Agent

Application Agents are agents that process requests by coordinating sub-requests sent to other agents. Typically this means parsing a single request (sent to the Application Agent) into several sub-requests which are passed to other agents, the application agent then does some of its own calculation, and passes the result back to the original requester. If the community is privacy controlled, then part of this calculation will be filtering responses according to the purposes in the Pluto session. Application agents require interface methods for their purpose, and they also typically need access to a directory service in order to find agents to handle sub-requests. These types of agents are not mutually exclusive, hybrid agents that are combinations of these types of agents are expected. An application agent might maintain its own list of simple agents, and act as a hybrid Directory/Application agent for example. These agent types are helpful in classifying agents, and understanding the interface requirements of an agent.

D. Inner-Community Co-operation

By itself, the basic architecture has several benefits, but this architecture is also designed to take advantage of the possibility that agents could exist in multiple communities at the same time. In order for an agent to belong to a community, it has to register with that community's Naming Service Agent and it has to adapt an interface that the community understands.

Registering with a new naming service is fairly simple. In order to register, the agent must have a unique name for that community and an address. Assuming these two criteria can be met, the Naming Service can add it to its list of agents in that community. In Secure or Privacy controlled communities, this will be complicated by the need to exchange private keys and permissions.

The interface requirement is usually more difficult to satisfy. There is no reason to assume that all communities will have similar requirements, so there may be some non-trivial work. Typically, there are two solutions. The first is using generic interfaces. The possibility of sharing interfaces across multiple communities is, after all, the reason why generic interfaces exist. If two communities expect the same generic interfaces from their agents, then adding an existing agent to a new community is simply a matter of notifying the Naming Service Agent in the new community. The other option is that new interfaces be added to existing agents. The agent interface is kept separate from the agent logic, so new interfaces should have minimal impact on the actual agent logic. There may be some new logic required, but agents are designed for a particular purpose and moving into a new community shouldn't change the agent's purpose. Because the purpose is unlikely to change, the majority of the logic should remain intact. Adding an agent to a new community should require, at worst, creating a new interface, that the new environment understands, and reusing existing logic.

3. E-commerce Communities

Once registered with the system, users log on to the e-commerce e-community using a web browser. The system hosts a ***user profile agent*** for each user that stores user interest information in a hierarchy. This profile is transparent to the user and is created automatically, but the user does also have complete control of what it contains and can set each interest to be private, restricted, or public. In the case of private interests, no other community member (buyer, salesperson) knows that the user has such interest. On the other hand, users can share public or restricted interests with other e-community members. Customers with common interests may open communication channels to share the shopping experience. ***Adaptive personal agent*** is an ideal solution for finding a user's personalized information. Because these agents can initiate tasks without explicit user prompting, they can undertake tasks in the background, such as searching for information. Since agents learn from experience, their knowledge of an individual increases over time, leading to improved accuracy of community data, including information about goods, customers, and contacts. In addition, by sharing their domain's public knowledge with other agents, they contribute further to the overall community knowledge. Another type of agent, the ***Contact-finding agent***, can locate members with distinct interests or competencies so that users can find experts in a given sub-domain or other members with interests similar to their own. Lastly, ***Collaborative-filtering agents*** specialize in promoting interaction among community members, allowing sharing of information among those who share the same interests.

4. Conclusion

Electronic commerce is becoming a major component of business transactions. With the creation and use of a collaborative commerce environment, the users can experience more and more functionalities that they encounter in a real-world shopping. The work presented here has significant impact on the practical applications of intelligent-agent-based e-communities of buyers and vendors in the industry.

Acknowledgements

The authors acknowledge the financial support of IBM Canada and the Ontario Research Network for Electronic Commerce.

References

[1] J. Marathe, "Creating Community Online", Durlacher Research Ltd, 1999,

[2] A. Warms, J. Cothrel, T. Underberg, "Return on Community: Proving the Value of Online Communities in Business", Participate.com, April 12, 2000.

[3] X. Shen, T. Radakrishnan and N.D. Georganas, "vCOM: Electronic Commerce in a Collaborative Virtual World", J. of Electronic Commerce Research and Applications, Vo.1, No.3-4, Aut.-Winter2002, pp. 281-300.

[4] Y. Zhang, L. Guo and N.D. Georganas, "AGILE: An Architecture for Agent-Based Collaborative and InteractiveVirtual Learning Environments", Proc. IEEE Globecom 2000 Conference, San Francisco, 2000.

[5] T. Finin Et al, "KQML as an Agent Communication Language". Proc. of the Third International Conference on Information and Knowledge Management (CIKM), ACM Press, November 1994.