Enhancing e-Commerce with Intelligent Agents in Collaborative e-Communities

Xiaojun Shen[1], Shervin Shirmohammadi[1], Chris Desmarais[1], Nicolas D. Georganas[1] and Ian Kerr[2]
[1] *Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER)*
School of Information Technology and Engineering (SITE)
[2] *Faculty of Law*
University of Ottawa, Canada
E-mail { shen | chrisd | shervin | 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. This architecture not only applies agent technologies in eCommerce system in novel manners, but also incorporates privacy law and legislation into its technical design, and in that respect it is different from other existing e-Commerce systems.

1. Introduction

Many existing e-commerce applications only provide users with a relatively simple, browser-based interface to access available products and services, which 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.

Traditionally, the term ***community*** refers to a location where people with common interests gather to ask questions, collaborate, or share social norms and experiences. Because they are present in the same locale, members can meet often to learn from each other by sharing their explicit knowledge and revealing information about their successes and failures. These communities use web technologies as a vehicle for disseminating knowledge and information quickly and inexpensively as well as for global communication and collaboration. Like traditional communities, e-communities act as repositories of information for their members. But what is better with e-communities is that they can store a larger amount of important data. 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[5][7].

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. Given the importance of collaboration and community, the architecture that we aim to develop will be founded not only on sound technologies but also on a carefully considered legal infrastructure. Our goal is to blend technology and law; to fuse software code with legal codes. In addition to creating an agent system that better reflects the world of commerce, we aim to create an e-commerce environment that respects ethical and legal notions such as informed consent and the protection of personal privacy. In other words, the agents adhere to basic consumer protection and privacy principles and follow very carefully existing ethical and legal norms.

The rest of this paper is organized as follows. In section 2, a multi-agent system for collaborative commerce implemented over Microsoft .NET framework is proposed. Section 3 discusses the privacy management in the e-community, while section 4 depicts the design and implementation of eCommerce communities over the proposed MAS system. The collaborative commerce aspect is focused on in Section 5. 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 the 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 in [3][6], which 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* between 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.

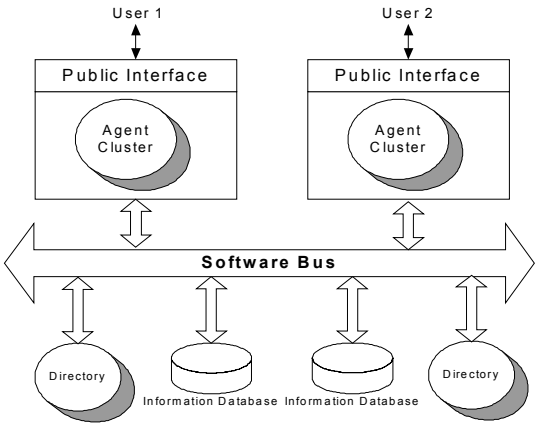

Figure 1 Generic Architecture of Agent-Based Collaborative Commerce

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.

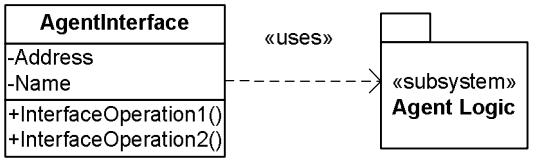

Figure 2 Agent Overview

Address

The address property is used to locate the agent in the distributed environment. The proposed system is implemented over Microsoft .Net framework, and in that environment the address was 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 a simple 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.

Interface

The interface property is what allows other agents or external applications to communication 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) [11]. KQML provides performatives to define the kind of interactions a KQML-speaking agent can have. A KQML message consists of three layers: the *content* layer, the *message* layer and the *communication* layer. The content layer bears the actual content of the message in the agent's own representation language. The communication layer encodes a set of message features, which describe the lower-level communication parameters, such as the identity of the sender and recipient, and a unique identifier associated with the communication. The message layer is used to encode a message that one application would like to transmit to another. The message layer forms the core of the KQML language, and determines the kinds of interactions one can have with a KQML-speaking agent. The performatives of a KQML message include those to request that an agent perform a task (*ask-one*), to provide other agents with certain information (*tell*), to watch another agent for a particular condition (*monitor*), and to register capabilities with another agent (*advertise*).

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 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.

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 find and contact resources in a community. The exact mechanics differ according to community needs.

Broker: A broker is simple name lookup service: an agent will send a name for an application and receive exact connection information (http address, machine address, port number etc.) so that the agent can connect to that service. The details are often passed as clean text.

Kerberos Server: Kerberos is a strong authentication protocol designed for client/server architectures [13]. The premise being that instead of returning simple connection details, a Kerberos server returns a ticket that is encrypted with details authenticating the exact user who can use the ticket and some simple restrictions on the agent's use of that ticket. A ticket has two parts: a client part and a server part. Both parts contain restriction information on the connection, and both parts are encrypted using private keys known only to the server or agent. Kerberos tickets allow an agent to authenticate itself to an application and create a private key that both parties can use to exchange secure information.

Pluto Server: As part of the architecture, a new protocol called Pluto was developed which integrates Kerberos and extends the ticket to include purpose information. Purpose information, discussed in more detail below, is a key requirement to ensuring privacy control in information flow. Pluto has been designed specifically to allow the exchange of private information in an e-community.

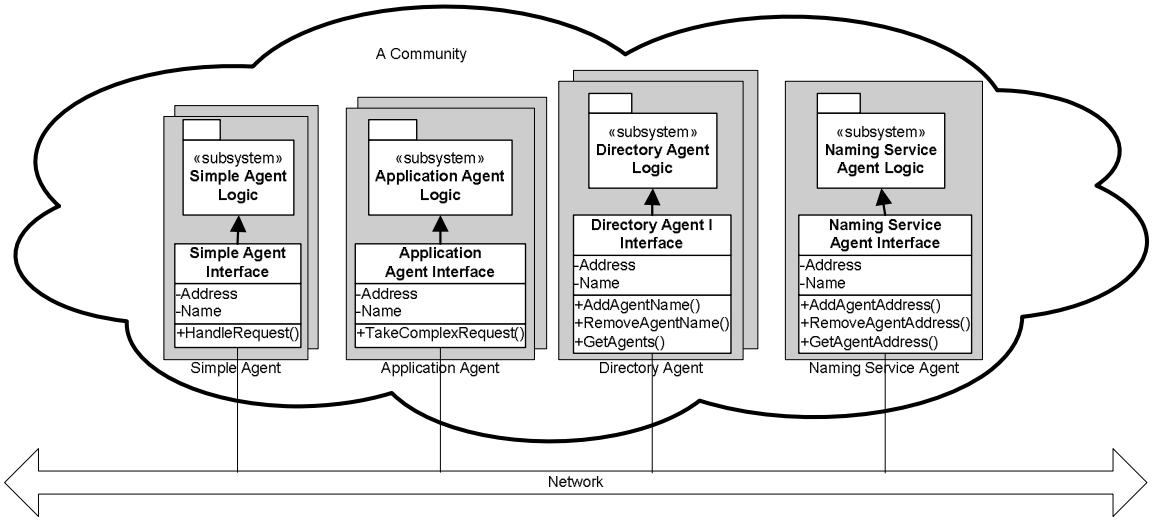

Figure 3 Agent-Based Community

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. Ultimately, the difference between a directory and a naming service is that a naming service is a complete naming system for an entire universe, where a directory is a much narrower view of related services. A naming service is a global resource; a directory service is a much more local grouping of related agents that are grouped for a specific purpose.

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. Privacy Management in Communities

In order to achieve an agent's purpose, it is expected that some agents will require some personal information about the customers (or other objects). Because the collection, use and disclosure of personal information can have legal consequences, a community of agents will be required to manage personal information in an appropriate way. Agents are capable of taking independent action, and if part of that action is, for example, the sharing of personal information, then agents must be designed to share personal information in a way that complies with legal privacy obligations.

The Code [12] is incorporated into Canada's comprehensive private-sector privacy legislation titled: the Personal Information Protection and Electronic Documents Act (PIPEDA). The Code and PIPEDA outline requirements for maintaining private data for organizations in Canada, and are being used to provide guidance in the design of privacy aware agents. Similar legislations exist, or are underway, in other countries. While the principles outlined in the Code and PIPEDA are important considerations for an e-commerce community, not all of them are particularly interesting from an agent architecture point of view. This paper will therefore look only at the principles of Safeguards; Accountability; Identifying Purpose; and Limiting Use, Disclosure, and Retention because those principles do impact the design of agent communities.

A. Safeguards

"Personal information shall be protected by security safeguards appropriate to the sensitivity of the information." [12, Principle 7]

In order to handle private data, an agent or community of agents is required to ensure that the system protects its information. The use of digital signature and encryption technologies can help ensure that agents in a system are actually what they claim to be, and to help ensure that personal information in a system can not be captured while in transit. The Kerberos protocol can ensure that all messages passed across a community network are encrypted appropriately, but it is not sufficient. Even assuming that reasonable security is achieved for passing personal information, there are still interesting questions in regards to controlling what information can be shared, and who it can be shared with. In order to control information for privacy, *purpose* information needs to be used for filtering.

B. Accountability

It is incumbent on organizations to specify which individual (or group) is responsible for managing personal information within an organization. The need to have identifiable responsibility is one of the forces behind the grouping of agents within a community. Because the responsibility for the flow of personal information between agents must be assigned to an individual (or small group), agent based systems should be designed to minimize the complexity of information flow. A privacy officer can then be assigned responsibility to manage both how private information is passed within a community, and restrict passing of information outside the community.

C. Identifying Purpose

In order to assure that an organization does not use personal information it is not entitled to, there are several principles that detail how information is to be gathered and maintained. When personal information is gathered it must be gathered for a specific purpose, and generally a person must give consent for that information to be used by an organization. Obviously this impacts how information is entered into a community of agents, but for the most part that is outside the scope of this discussion. The important part, from an architectural point of view, is that personal information in a community must have an identifiable *purpose* linked to it.

In non-agent based applications this is usually a trivial requirement because most applications have fairly simple purposes. The fact that information exists in an application can normally safely infer that the information was collected to be used by that application. In an agent-based architecture, however, agents are expected to have their own individual purposes. There is no requirement for an entire community to have a single purpose. Instead, agents

using personal information must maintain two sets of information: the personal information, and the purpose information (information about the scope and purpose for having the information in the system). An excellent example of how this information might be kept is contained in the P3P standard [14]. P3P is a client/server privacy solution that uses XML to store and pass privacy filtered information. Part of the P3P protocol is a data schema for describing data and the policies for which that data may be released.

D. Limiting Use, Disclosure, and Retention

The reason purpose information is maintained by an organization, is that private information "shall not be used or disclosed for purposes other than those for which it was collected" [12, Principle 5]. This principle becomes important because it allows agents to make reasonable decisions about what information can be passed between agents. If an agent (the requester) requests data from another agent (the provider) in a system, then the provider must ensure that the privacy of its information is maintained. The provider can compare the requester's purpose to the data's purpose. If consent has been given for the requester's purpose, then the data can be passed to the requester. The requester's purpose and consent do not have to be a perfect match, it is possible that a requester's purpose might be "reasonably related" to the consented purpose (as covered in the 12, Principle 4.3.5). Agents can then be designed to make good comparisons between the purpose of other agents and the purpose of information.

E. Privacy Aware Community

In order to have agents that maintain personal information (data agents), they need to keep both the personal information and scope/purpose information about what consent has been granted for the use of that information. Any agent that wishes to use personal information (functional agents) will be required to maintain information describing the purpose of the agent. A final agent, privacy policy agent, will maintain a semantic web that relates purposes, as shown in Figure 4. When a functional agent requests personal information from a data agent, it includes its purpose as part of the request. The data agent can then make a request of the privacy policy agent to determine what data it can legally/ethically transfer to the request agent. The privacy policy agent will make this determination by searching for a path between the scope/purpose information about the data, and the purpose for the proposed disclosure of the personal information.

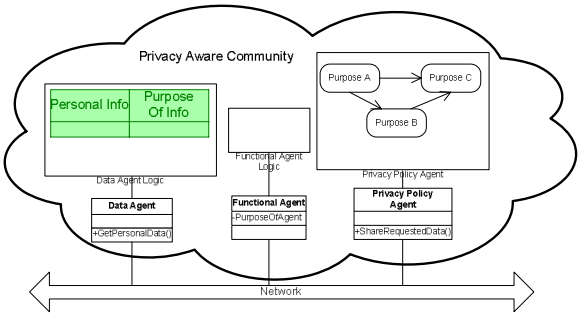

Figure 4 Privacy Management

4. e-Commerce Communities

A. User Agent Cluster

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.

B. Voice-enabled Assistant Agent

A voice-enabled assistant interacts with the customer using voice synthesis and helps him/her navigate efficiently in the e-Community.

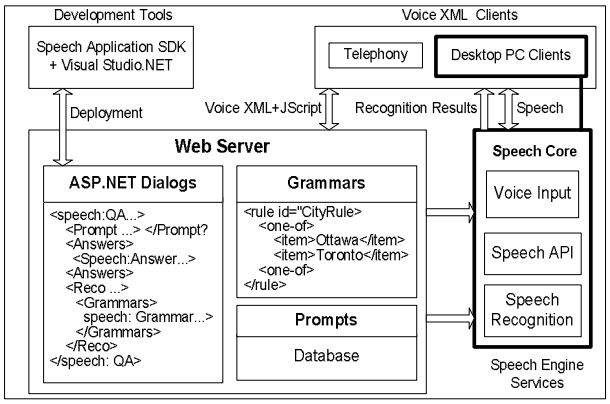

Figure 5 Implementation of Voice-enable Assistant

Figure 5 depicts the implementation of voice-enabled assistant with Microsoft Agent [10] and Speech API technology. VoiceXML[9] is used to define dynamic ontology. Voice-enabled agents provide users with a user-friendly speech-command interface, acting as a general "help" facility for the user by accepting simple voice commands and giving voice responses. This provides a more natural interface for users.

5. Collaborative e-Commerce

The design and implementation of current online shopping places have primarily focused on the process of exchanging goods. Online catalogues of mail-order companies are created and metaphors of shopping baskets and virtual cash desks are introduced. While these metaphors aim at easing the process of shopping by emulating real world experiences, current virtual market places often lack in the emulation of the social interaction factors.

We believe there is need to combine the virtual market with the social place again. Customers who participate in the virtual market should change their role from consumers to people who want to satisfy their wide range of needs through shopping. The purchase of goods is only one of them; social interaction, learning, or excitements are others, which can be satisfied in a community. The role of markets that bring together people, who did not know each other, could create new social communities.

To validate the proposed architecture, an e-commerce community, involved with sale of bedding, is designed. In this community, businesses might be related to the sale of mattresses, pillows and sheets, and customers who wish to purchase those products. There would be special purpose application and directory agents to co-ordinate the tasks of purchasing. Each individual business would create it own sale agent inside the community responsible for the sale of its products. Those sales agents might make use of business' proprietary community, or might contain all the logic to handle sales for that business. Customers could also have agents that would handle purchasing for that customer. It would keep track of desires and preferences of a specific customer, and be given the authority to act as that customer's proxy inside the system.

Customer-Business Interaction (C-B)

The obvious interaction between agents in this system is for a customer agent to buy a product from a Business Sales Agent. A directory agent can be used to maintain a list of all sales agents that sell products. A customer interested in buying that product can then get a list of all sales agents for a product from that directory agent, and place an order with an appropriate sales agent (Figure 6). The next section will discuss the negotiation process between customer and sale agent in details.

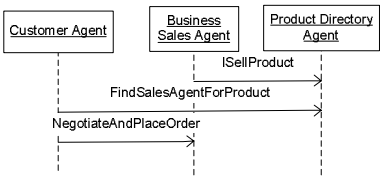

Figure 6 Customer - Business Interaction

Customer-Customer Interaction (C-C)

A more interesting scenario would be to group customers, so that they could to take advantage of group rates offered by some businesses. A separate application agent could keep track of customer agents requesting a product, and when there are enough customers interested in a product it could make a joint sale (Figure 7).

A more interesting scenario would be to group customers, so that they could to take advantage of group rates offered by some businesses. A separate application agent could keep track of customer agents requesting a product, and when there are enough customers interested in a product it could make a joint sale (Figure 7).

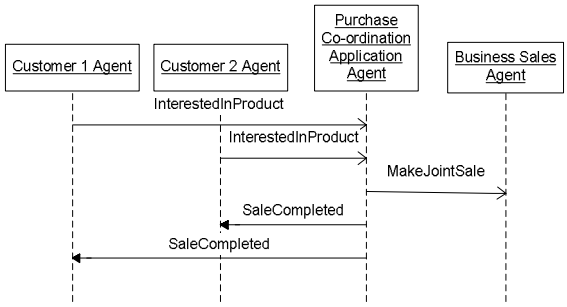

Figure 7 Customer - Customer Interaction

Business-Business (B-B)

Another interesting interaction might be to allow businesses to group their products into a bundle deal. A single sales agent could sub-contract parts of a sale to other businesses in the community (Figure 8).

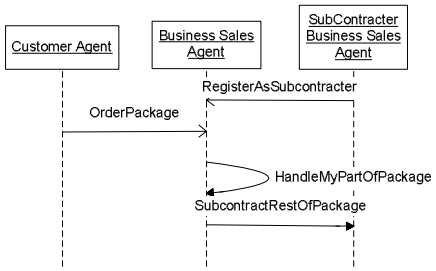

Figure 8 Business - Business Interaction

6. 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. Many current e-commerce applications exploit agent technologies to misrepresent online products and services or to surreptitiously gather and mine personal data [8]. By focusing on consumer protection and privacy principles as a significant design feature, a value-centered design process was created so that important policy and legal values are preserved; recognizing that respecting end-user privacy in fact makes good business sense. Moreover, ontology-based approaches for the abstraction of generic interface will be investigated in the future, which are well-known to promote interoperability among heterogeneous parties.

7. Acknowledgements

The authors acknowledge the financial support of IBM Canada, and the Ontario Research Network for Electronic Commerce, as well as the technical development efforts of Xingyuan Wang, and the legal studies and legislation research of Alex Cameron.

8. 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] 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.

[4] G. E. Kersten, G. Lo. “Negotiation Support Systems and Software Agents in Business Negoitaions,” The First International Conference on Electronic Business, Hong Kong, 2001.

[5] 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.

[6] L. Guo and N.D. Georganas, “Towards Agent-based Collaborative and Interactive Virtual Environments”, Chapter 6 in “Virtual Reality Technologies for Future Telecommunications Systems", Willey, 2002.

[7] J.C. Oliveira, X. Shen and N.D. Georganas, “Collaborative Virtual Environments for Industrial Training and e-Commerce”, Chapter 7 in "Virtual Reality Technologies for Future Telecommunications Systems", Willey, 2002.

[8] I.R. Kerr, “Bots, Babes and the Californication of Commerce”, U. of Ottawa Law and Technology Journal, 2004.

[9] Voice Extensible Markup Language (VoiceXML) Version 2.0 http://www.w3.org/TR/2001/WD-voicexml20-20011023/

[10 Microsoft Agent, http://www.microsoft.com/msagent

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

[12] Department of Justice Canada, The National Standard of Canada Entitled Model for the Protection of Personal Information, CAN/CSA-Q830-96 http://laws.justice.gc.ca/en/P-8.6/92379.html

[13] Keberos , http://web.mit.edu/kerberos/www/

[14] P3P, http://www.w3.org/P3P/