SERVICE CENTRIC BROKERING IN DYNAMIC E-BUSINESS AGENT COMMUNITIES

Sumi Helal Computer and Information Science and Engineering Department
University of Florida
helal@cise.ufl.edu

Mei Wang
Computer and Information Science and Engineering Department
University of Florida

ABSTRACT

The paper focuses on the development of protocols for brokering-based agent communities in e-Business. The global market is driving e-Business to become service-centric, offering highly modular e-Services that can be flexibly and dynamically composed into rapidly deployable e-Businesses. This trend is giving rise to a new set of requirements of negotiation-based, autonomous, and intelligent computing. It is, therefore, expected that in the near future, e-Services will be designed and implemented as software agents (also known as agent-based systems). This paper prepares for the proliferation of agent-based systems in e-Business by contributing a suite of protocols for self-organizing agent communities. The protocols are based on a three-tier architecture of agents, brokers, and superbrokers. We present the architecture and the protocols (*Broker-Based Agent Community Protocol*s, or BBACP). An implementation using *JKQML* is also presented along with a case study drawn from the electronic auto-trading domain.

1. Introduction

Conducting business electronically is not a new idea anymore. The development of e-Business has been observed in several ways: flexibility requirement in the business market, emerging virtual enterprise or net enterprise, and business globalization [Ouzounis & GMD-Fokus 1998, Zimmermann 1998]. More and more small or medium size enterprises or private households are connected with each other through the Internet. However, such web-based e-Business applications and systems developed by individual companies are neither compatible nor interoperable with each other. New mechanisms and technologies need to be invented in order to make individual standalone e-Business interoperable and cooperative. In our view, agent technology is one of the core technologies that will accommodate the growing proliferation of e-Business.

Agent technology has been widely adopted in the artificial intelligence and computer science communities. An agent is a computational system that operates autonomously, communicates asynchronously, and runs dynamically on different processes in different machines, which support the anonymous interoperation of agents. These qualities make agents useful for solving issues in information intensive e-Business, including speaking ontologies, advertising, service exchange, and knowledge discovery, etc. In a fast changing e-Business environment, the interoperation and coordination across distributed services is very important. The desire for more cost efficiency and less sub-optimal business processes also drives the employment of agent technology in e-Business.

With the support of agent technology, more e-Business agents will appear on the Internet providing e-Services as well as exchanging information and goods with other agents. The interoperation of e-Business agents leads to the formation of the e-Business Mall, which is an interaction space of agent communities under various business domains.

The significant problems in the e-Business are the information deficiency and asymmetry between the business participants. It is also difficult for each participant to exchange information products and services in an efficient manner, and to partner in a virtual enterprise. We contribute the concept of agent community, in the context of e-Business as an approach to those problems.

An e-Business agent community is a self-organized virtual space consisting of a large number of agents and their dynamic environment. Within a community, highly relevant agents group together offering special e-Services for a more effective, mutually beneficial, and more opportune e-Business. Each agent community consists of agents specializing in a single domain/sub-domain, or highly intersecting domains.

The ideal e-Business agent communities aim to

- make it easy to develop virtual enterprises, on which companies are joining and sharing resources and business processes for the production, marketing, and other services, provide protocols for the exchange of information, products and services among business entities
- represent certain potential relationships among potential members, and support the interoperation among them
- help in the establishment of relationships among agents with common interests
- provide discipline in the open network in anticipating of e-Business proliferation
- accommodate solutions to issues such as trust, efficiency and credentials

In section II, we present the layered architecture of agent communities and describe the specification, analysis, and design of agent organizational structure. Section III addresses the role of ontologies in agent communities. Section IV presents the components of the *Broker-Based Agent Community Protocol*s (BBACP) in detail. The protocols or rules in the service-centric *Brokering Layer* include processes of joining e-Business in the community, knowledge discovery and exchange, and advertisement of services or capabilities. In section V, we illustrate how the community and the supporting protocols are designed and implemented using the *JKQML* (http://www.alphaworks.ibm.com/formula/jkqml) agent shell. We present the functionality of the community components and our implementation of the layered community architecture and protocols as Java APIs supporting interoperation within an agent community. Finally, an e-Business case study based on the BBACP protocol is demonstrated, showing the benefits of service-centric brokering in self-organized agent communities.

2. Architecture of Agent Community

In order to meet the challenges in the developments of future *e-Business Communities (Mall*s), a loosely coupled architecture with heterogeneous components is required for different e-Businesses. We introduce a hierarchical brokering architecture for the e-Business agent community to support the appropria level of flexibility, scalability and interoperability in e-Business as well.

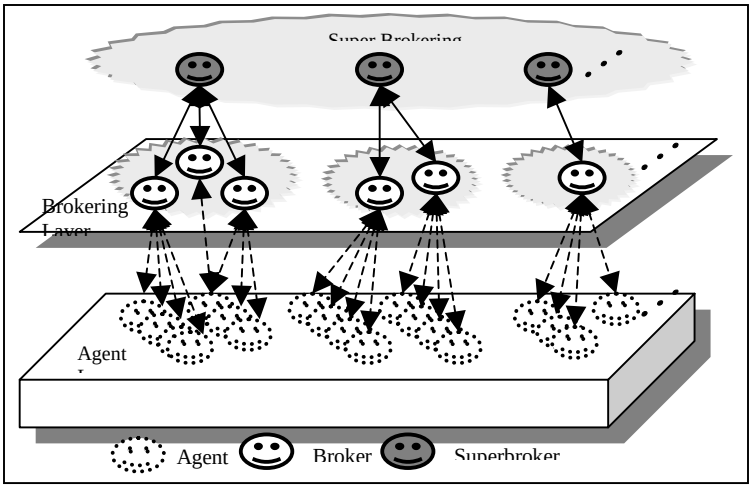


Figure 1. Hierarchical Architecture of an Agent Community

As indicated in Figure 1, an agent community is composed of an *Agent Layer*, a *Brokering Layer* and a *Super Brokering Layer*. The *Agent Layer* is the bottom layer in the community architecture hierarchy, where each agent or agent based system (ABS) provides specific e-Service(s) and inter-operate with each other. In the e-Business environment, service-centric agents self-organize together to exchange information or product under a certain knowledge domain. The business processes among agents are completed under the facilitating of a special agent, a broker. A broker represents and facilitates a set of agents with similar e-Services and common interest in the e-Business community, which forms a brokered agent system.

The autonomous e-Business agents organize together based on the specialization of an individual broker that represents the best interest of a set of e-Services. With the broker's facilitating, information and services exchange among heterogeneous agents becomes efficient. Agents communicate their needs and capabilities toward brokers.

Brokers use the acquired agent knowledge to reply or rout messages to the appropriate agents. A brokered agent system provides a dynamic structure as agents can be introduced and removed at runtime. Moreover, a brokered agent system can be built from a set of heterogeneous agents, which assists in creating a very flexible coordination in a loosely coupled system. We assume that a useful domain ontology exists and is being shared by agents in the brokered agent systems.

In the *Brokering Layer*, the interoperation of various brokers forms a multi-brokering system, which consists of multiple brokers and a superbroker. A superbroker is a well-known broker in the multi-brokering system, supporting the knowledge sharing efforts among registered brokers. A broker advertises or unadvertises e-Services to the superbroker it registers with. Usually only the service requests that can't be satisfied within a brokered agent system are sent to other brokers within the community. The capable brokers in the community provide the requester with the e-Service, which is analogous to the expansion of the knowledge of requesting brokers. A superbroker maintains the service subscriptions from its members and notifies the subscribers as long as the right capabilities are available in the community. In the mean while, individual brokers maintain their own autonomy, facilitating interaction of self-organized member agents. In a multi-brokering system, brokers operate under a common protocol, the *Brokering Protocol*, that governs how they advertise capabilities and services of agents they represent, how they reason and select which broker is appropriate for the requested services, and how they communicate and coordinate over community common ontologies.

On the top of the hierarchy is the *Super Brokering Layer*. In this layer, superbrokers interact with each other directly or through superbroker consortia, which eventually leads to the concept of virtual enterprises and *e-Business Mall*s. In this layer, superbrokers operate under a common protocol, the *Super Brokering Protocol*, that governs how they communicate and coordinate over common ontologies, how they reason and solve the community issues, such as trading brokers or other community restructuring.

The community is easily scalable by accepting new brokers that represent a set of agents. When the new set of agents represented by a broker joins the community, the superbroker becomes the entry point for interaction to the new broker. The minor communication overhead also ensures the necessary scalability for the agent community. Upon a new broker joining or leaving the community, the change of the knowledge and services will be updated in an information repository maintained by the superbroker and known to each of the existing member brokers. Superbrokers have tight control over load balancing, support great service reliability, and provide potential wide ranges of services in the e-Business community.

Researchers have designed different organizational structures for brokering systems, such as the peer-to-peer multi-brokering architecture in *InfoSleuth* [Nodine et al. 1999], the partitioned repository design in blackboard systems [Carver & Lesser 1992], centralized message routing design in *JAT Lite* (http://java.stanford.edu/), and the multi-facilitator mediation in *OAA* [Martin et al. 1999]. Comparatively, the hierarchical agent community is designed to meet the requirements of the proliferation of e-Business agents, to accept new brokers without the direct necessity of adjustment on the Agent Layer. In the Brokering Layer, brokers communicate with each other directly or indirectly under the superbroker supervision. The communication and interaction protocols specified in Broker-Based Agent Community Protocol are capable of avoiding a single point failure, not like those in *JAT Lite* and blackboard systems. What's more, it is obvious that the knowledge of all other brokers is not necessary. Each broker only needs to understand the superbroker it registered with, and communicates with some other brokers upon necessity. Thus the amount of messages flowing among the brokers within the community is minimized, providing the solution to the network bandwidth restriction imposed on various e-Business applications.

3. The Role of Ontology

In order to self-organize into communities, each broker should contain a generic broker instance, which implements at least the basic infrastructure for an agent community. Some assumptions are necessary to the joining protocol:

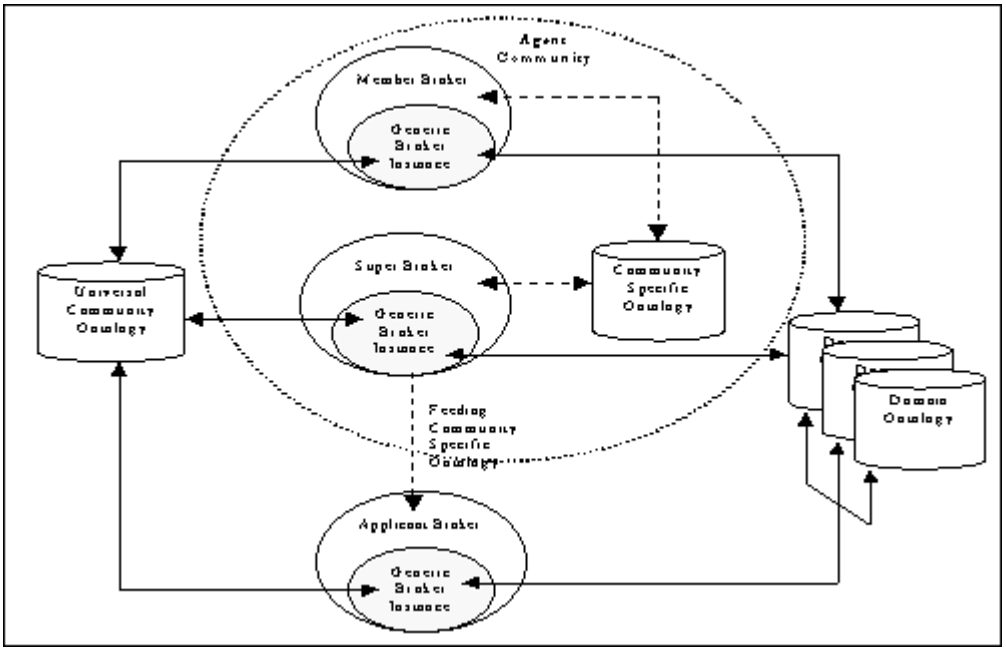


figure 2. Universal Community Ontology, Domain Ontology and Community Specific Ontology

Each generic broker instance has a *Universal Community Ontology* and can access various publicly available *Domain Ontologies* (Figure 2). *The Universal Community Ontology* is the fundamental ontology applicable to all e-Business communities, which provides brokers the minimum amount of vocabularies and knowledge toward self-organizing features. The *Universal Community Ontology* enables brokers to interact with a superbroker, conveying the spoken ontologies, and knowledge capabilities among other information. A *Domain Ontology* provides vocabularies and knowledge within an e-Business sub-domain.

In each e-Business community, the *Community Specific Ontology* should be made available and open to both community members and applicants. Thus, prospective community members are able to browse and learn about the available *Community Specific Ontology* maintained by the superbroker before initiating any application. Based on *Community Specific Ontology*, the applicant broker can determine the relevancy between its own knowledge and that of a community, as well as other self-organizing factors. Only if knowledge relevancy and value added are expected, would further procedures need to proceed.

The *Universal Community Ontology* and *Domain Ontologies* possessed by the applicant broker enable it to initiate an application procedure. Applicant brokers have the goals of learning more about the *Community Specific Ontology* and other community facts. The superbroker maintains the *Community Specific Ontology* and acknowledges them to the applicants during the joining procedure. The joining procedure occurs when an applicant broker sends a membership application to the superbroker. In a service-centric agent community, membership application presents its knowledge as a constrained sub-domain of the e-Business domain. By using the *Universal Community Ontology* and *Domain Ontology*, the broker provides the general information about itself, including basic information (e.g. name, location, status) and capabilities (e.g. knowledge of a sub-domain, e-Business transaction types, service credential characteristics). Currently, the *Universal Community Ontology* is implemented and described using KQML [Finin et al. 1997] as low-level language.

4.Protocols for Agent Communities

We introduce the *Broker-Based Agent Community Protocol*s (BBACP) to enable inter-organizational business procedures. The BBACP consists of an *Agent Protocol*, a *Brokering Protocol*, and *a Super Brokering Protocol*, which provide mappings from states to action in each agent community layer.

The *Agent Protocol* involves interaction with other agents, which varies so as to suit to the business application and usage scenarios. Each agent-based system follows a local protocol, the *Agent Protocol*. A local protocol is a function from local states to local actions, such as applying rules or sending messages [Ouzounis & GMD-Fokus 1998]. Therefore the *Agent Protocol* is the base-level commitment and responsibility to the community. Because it is an open protocol, much space is left for negotiation when agents are built. The autonomy of the agents themselves

also requires self-government and self-organization. By accepting the *Agent Protocol*, agents commit themselves to play special roles and to guarantee properties such as sincerity and producing appropriate responses [Huhns & Singh 1998].

The *Brokering Protocol* extends the brokering function to meet the needs of the interoperability among the brokers and with the superbroker. The *Brokering Protocol* describes a cooperative multi-brokering system, which provides the solution to interoperation in a dynamic heterogeneous agent community. Each broker performs basic brokering functionality (such as service discovery and knowledge sharing) within a set of e-Business agents, as well as representing e-Services in the community. A superbroker is on the top of the multi-brokering hierarchy, plays the monitoring role in the agent community and requires the registration of member brokers. Individual brokers representing a set of agents are allowed to advertise their business service and send capability queries to other brokers, as well as receive advertisements from other brokers. In addition to regulation of joining in and leaving from the community, other issues covered in the Brokering Protocol include business knowledge discovery and sharing, information load control, and brokering monitoring.

The *Super Brokering Protocol* governs how a superbroker communicates and coordinates in the e-Business Mall, how they reason and solve the community issues, and how to form and maintain the superbroker consortium. Our research focuses on the *Brokering Protocol* layer, which includes the following components.

4.1. The Joining Protocol

The protocol of joining a community implements a logic conversation between brokers and superbrokers. The conversation is a sequence of rule-based communications. Conversation policy (or rule) specifies the social behavior of brokers and superbrokers at each state. The conversation policy is mapped into a *Finite State Machine* (FSM) as illustrated in Figure 3.

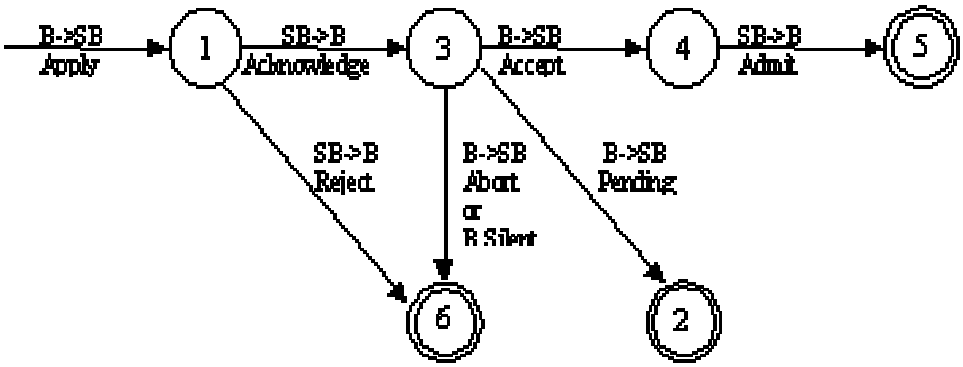


Figure 3. FSM for Community Joining Protocol

Each broker declares to the community what kind of ontology concepts it can deal with, enlisted in its application to the superbroker. After receiving a registration attempt (State 1), the superbroker conducts multi-faceted membership evaluation of the applicant, such as knowledge and capability relevancy, creditability, possibility of value-added, potential expansion, among others. The conclusion of the evaluation is either reject (State 1 -> State 6) or acknowledge (State 1 -> State 3). The acknowledgement message contains three components: *Domain Ontology*, *Community Specific Ontology* and *Bylaws*.

At State 3, the broker received the acknowledgement and automates its behavior related to ontology adjustment or other knowledge changes.

If a positive conclusion can be made immediately, a message notifying the acceptance of *Domain Ontology*, *Community Specific Ontology* and *Bylaws* will be sent to the superbroker (State 3 -> State 4). The finite state (State 5) is reached when the superbroker confirms the grant of membership upon receiving the acceptance message.

If a negative conclusion is made, an abort message will be sent to the superbroker (State 3 -> State 6), which indicates the end of the conversation resulting in the finite state (State 6) being reached.

If a decision can not be made, a pending message is sent to the superbroker. In order to allow the applicant to continue the member application in the future, the superbroker must enforce the knowledge of access history. It is clear that the state information should be kept and used in guiding conversation policy. The question is where the state information should be kept. Our approach is to embed state information in the conversation policy in the superbroker side. During the application processing in the future, the superbroker will check the applicant accessing history, thus following the conversation policy based on state information.

If no replies are received by the superbroker within a specified period of time, the synchronous conversation will move to the finite state of application failure (State 6).

Upon approval of a successful registration, the superbroker updates the new broker in its repository. Failure of registration could encourage broker to engage in additional learning activities. Thus the superbroker manages the relationship between ontologies and brokers, and maintains ontology consistency among members.

4.1. The Departure Protocol

The unregisteration is used to inform the superbroker that the broker won't be able to provide the advertised service anymore. The leaving procedure is initiated by the leaving broker sending an "UNADVERTISE" message to the superbroker. The updated knowledge and services available to the community are "TELL" to all members. Alternatively, the leaving broker can broadcast the "UNADVERTISE" message to member brokers directly. The previous approach is implemented in the thesis. Upon receiving the unadvertsing message, superbroker and related brokers remove the services from their repositories. The obtained knowledge in the leaving brokers is removed as well.

Then the leaving broker sends an "UNREGISTER" message to the superbroker. Upon receiving the message, the superbroker removes the broker's entry from the member repository and marks it in the member credit system. The member credit system is the repository maintaining the history and service quality of previous and existing members. It acts as a quality monitor to ensure the knowledge sharing efforts in the community and prevent the malfunctions of community members.

4.2. The Knowledge Discovery and Exchange Protocol

The knowledge discovery is initiated by the inquiring broker, who multicasts the service request(s) to the community. Upon receiving the request(s), the capable brokers reply with appropriate service information that enables the requester to communicate with the service providers and obtain services from them directly. In our implementation, the inquiring broker sends a request to a superbroker. The superbroker searches its capability repository for service(s) that satisfy the request. If there exits at least one match, it sends the result as a "REPLY" KQML message to the requester. The received service information contains the required parameters to communicate with the service provider through KQML messages. Thus the requester is able to directly negotiate with the service providing broker(s) using specified language and ontology. The amount of knowledge discovery is based on the brokers' objectives: ASK-ONE (one capable service is recommended), ASK-ALL (a list of services are returned).

Subscription means that a broker requests to be informed if some specific service becomes available in the community, which is another approach of knowledge discovery. The subscription is presented in the performative "SUBSCRIBE". A broker may broadcast a subscription to the community so that each member broker stores it. If a broker is capable of serving the subscribed service, it will reply to the subscription and start a conversation with the message sender. Otherwise, the receiver does not response except for storing the subscribed subjects in its subscription repository. Once a broker, or an agent it represents, develops new capabilities that meet the specification of the subscription, the broker will notify the subscriber and initiate conversation. In our implementation, a superbroker acts as a special broker that maintains all the capability information in the community. Thus upon receiving a subscription, the superbroker searches the community capabilities repository to see if any services available in the community match the requests, and checks if the capable broker is up and running. Then, the query result is sent to the subscriber by using the "TELL" performative. If no knowledge is able to serve the request at the time being, the subscription is stored in the superbroker's subscription repository. If some new services satisfying the subscription are advertised in the community, the superbroker then notifies the subscribers of the availability of the new capabilities. With the received information, the subscriber can then obtain the service by initiating the conversation with the service provider(s). On that occasion, the knowledge sharing efforts are directly conducted among the subscribers and the service providers.

The unsubscription is used to inform the community that a broker no longer needs to be informed for the subscribed service. A broker can unsubscribe one service at a time by "UNSUBSCRIBE" performative. The "UNSUBSCRIBE" message is either multicast to all community members directly or sent to a superbroker and consequently sent to member brokers. In either approach, the receiving brokers remove the subscription from their subscription repositories. We implement the latter approach. When a superbroker receives an unsubscription message, it removes the corresponding broker's entry from the subscription repository so that no more notifications are sent to that broker.

Knowledge discovery follows knowledge discovery rules (a finite state machine). If a satisfied discovery is received, e.g. "TELL", the broker updates its acquired capabilities repository; if unsatisfied discovery, e.g. "SORRY", it could refine the request content and re-send it to the superbroker.

4.3. The Capability Advertisement Protocol

The advertisement is sent first to the superbroker by a broker who wants to advertise its capabilities to the community. The advertisement specifies the required KQML fields, which the interested broker has to use in order to be able to get the desired service. The performative is "ADVERTISE". If a broker wants to advertise multiple services, it does so by sending multiple advertisements. The advertisement should also be broadcast to the community by sending an "ADVERTISE" message to each member broker.

When it receives an advertisement, a broker updates its repository by adding a new entry for the new advertised services, inserting the name of the broker, the credential information, and domain dependent information. If the advertised service is new, or the advertisement exists and adjustment is made, then a "TELL" message is sent back to acknowledge the advertiser. Otherwise, a "SORRY" message is sent to inform the advertising broker about the failure.

The unadvertisement process is triggered whenever changes are made to any services provided by an individual agent. Upon receiving a service change message from its member agent, a broker updates its self-capability repository, which means either removing the entry permanently or updating the characteristics of this particular service. The broker then sends an "UNADVERTISE" message to the community to indicate the necessary changes needed to keep member's information repository up-to-date. Upon receiving the unadvertising message, all member brokers update their service information repositories.

In our implementation, a superbroker has much broader knowledge and maintains all the advertisements in the community. Services are advertised and unadvertised to the superbroker and interested members and then notified by the superbroker. Since the superbroker maintains the subscriptions from its members, the flow of advertising messages is reduced by only sending to subscribers and requesters instead of broadcasting to all community members. Upon receiving an advertisement, the superbroker first updates the entries in the capability repository. Then it checks the subscription repository to see if the advertised service satisfies any subscriptions. If one or more matches are found, the superbroker notifies the corresponding broker(s) by sending a "REPLY" message in which the service information is specified.

Upon receiving a service-changing message from its member agent, a broker sends an "UNADVERTISE" message to the superbroker to indicate the necessary changes needed to keep the information repository up-to-date. The superbroker updates the community capability repository by either removing the entry permanently or updating the characteristics of this particular service. Subsequently it sends notifications to member brokers who have acquired or subscribed to that particular service.

5. Implementation

Agents operate and interact in the infrastructure provided by the agent shell—*JKQML*. Figure 4 shows the design and implementation of our agent community architecture.

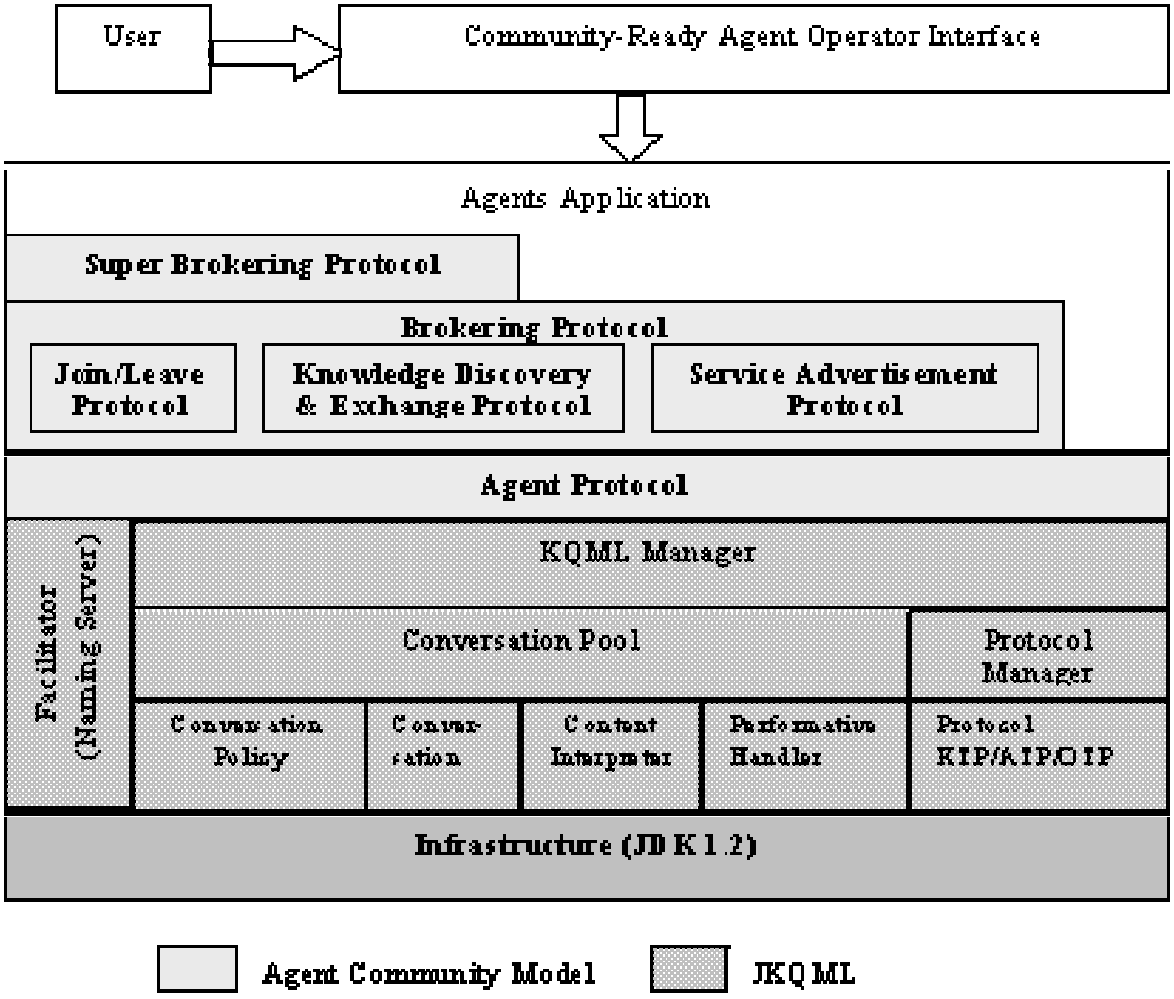


Figure 4. Agent Community Architecture Design and Implementation

The lowest layer is the *JDK 1.2* that compiles the Java source code into bytecode that can be executed in any open network through Java Virtual Machine. Above the Java infrastructure rests the Java-based KQML (*JKQML*) (http://www.alphaworks.ibm.com/formula/jkqml), the agent shell (or agent development toolkit). Developed by IBM, *JKQML* is a framework to construct KQML-speaking software agents, to build loosely coupled distributed systems by using an agent communication language—KQML [Finin et al. 1997], to provide flexible interoperability to exchange information and services between software systems. *JKQML* is written entirely in the Java language to construct software agents communicating over the Internet. It is designed to support various transport protocols, and to provide flexibility for the extension of the framework. As show in the figure, it provides interfaces to process KQML messages for agent application through the *KQML Manager*. The agent applications can access other major components such as conversation pool and protocol manager through these *KQML Manager*'s interfaces. Currently *JKQML* supports three transport protocols, one is the *KQML Transfer Protocol* (KTP) for the ordinary TCP/IP environment implemented in Java.net.Socket, the *Agent Transfer Protocol* (ATP) for the *Aglets* environment [Aglets 1999], and the *Object Transfer Protocol* (OTP) for transferring Java Object.

The agent community model defines how agents interact with each other in a cooperative manner. It utilizes and interacts with the *KQML Manager* and extends the functionality to support the BBACP, which is composed of three sub layers: *Agent Protocol*, *Brokering Protocol*, and *Super Brokering Protocol*. The *Agent Protocol* is the base-level commitment and responsibility to the community, which differs among different business providers. Because it is an open protocol, much space is left for adjustment when agents are built. The middle layer of the community model is the *Brokering Protocol* that provides the inter-brokering among community members. The upper level of the

community model is the *Super Brokering Protocol*, which extends the concept of agent community of virtual enterprise to the e-Business Mall.

We implement the Java classes such as *CRAgent*, *Broker*, *SuperBroker*, and *SocialViewer*, each of which captures one or multiple functions in the community. There are two inheritance relationships in Figure 5. The *Broker* class inherits the *Agent* class and the *SuperBroker* inherits all the functions of the *Broker* class. Inter-agent and inter-broker interaction is also illustrated.

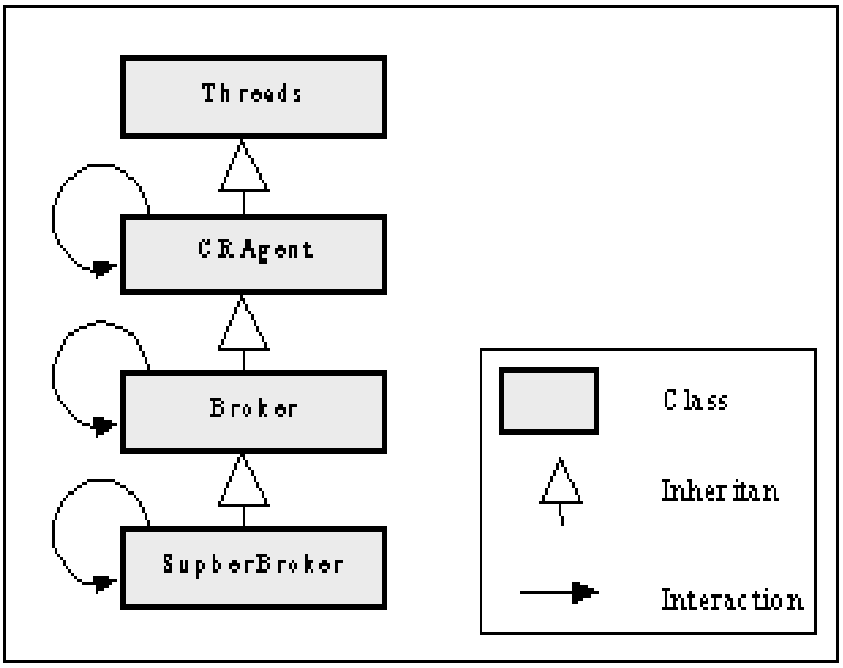


Figure 5. Community Components Diagram

Each broker, implemented as *Broker* class, maintains three information repositories: self-capabilities repository, acquired-capabilities repository, and subscription repository.

The superbroker, implemented as *SuperBroker* class, maintains the service information repository, guarantees relevance and affinity, enforces interoperations and knowledge sharing efforts, and tracks member brokers' credential characteristics. The superbroker can monitor the knowledge sharing efforts among members, provide efficient load balancing, and keep track of other credential information of each broker. The credential information repository contains the broker name, location, ontologies, status, cost of services, reliability, and quality. By maintaining credential information of member brokers, the superbroker can initialize a membership evaluation procedure. All members vote on the quality of every membership and submit voting results to the superbroker. Based on the voting result, the superbroker expunges the membership of unqualified broker(s).

The social viewer is a special agent that provides a domain-independent social view service. It is not an essential part of our architecture even through it views and analyzes social roles of agents, brokers, and superbrokers, shows their organizational inter-relationship, as well as indicates the message exchange and collective behaviors between brokers and superbrokers during their interoperations.

Our research focuses on the middle layer of community protocols, the Brokering Protocol layer. Our implementation provides community-ready interface, and supports the self-organizing feature in open e-Business communities. The community-ready interfaces are sets of APIs, allowing software programmers to easily write the agent applications that inherit the functionality of the organizational roles needed in the agent communities. The complexities and details of the protocol implementation are hidden from programmers. The APIs allow programmers to extend the classes and to overwrite the methods so as to implement more functions for specific business domains while maintaining the essence of brokered community protocols.

Appendix A summarizes the community-ready API for the BBACP protocols. It shows the API components of each layer and describes component interactions through KQML performatives. More detailed explainations of the API implementation can be found [Wang 1999].

6. Case Study

We demonstrate an application of the agent community and the community protocols to the on-line auto-trading domain. The objective is to enable the business participants to electronically buy and sell cars over an open network via e-Business agents.

To present the characteristics of the business process of buying and selling cars, we assume there are several brokers, one superbroker and a variable number of agents. The size and type of brokers are not limited to the description. As an open network, the auto trading community allows new brokers to join based on their knowledge relevancy and value-added to the community. Currently we designed a simple community consisting of participants such as: Individual Buying Broker, Dealing Broker, Individual Selling Broker, Manufacturer Broker, Warranty Broker, Auto Transport Broker, Vehicle History Broker, Auto SuperBroker, Whitepages and Yellowpages.

As described in the domain specification, all brokers automate the trading process among themselves. However, the direct business protocols are facilitated and maintained by a specialized broker, the AutoSuperBroker. The multi-brokering trading community contains several roles. It is possible for each individual roles to be played by an individual broker. Alternatively, a single broker can play several roles.

Each role played by a broker contains some responsibilities, such as sending registration in order to join the trading community, advertising its services or requesting one, and interacting with other type of brokers. This section introduces the role responsibilities for each broker including Member Applicant, Membership Inquirer, Services Advertiser, Service Inquirer and Subscriber and Broker. Tables 1 to 5 describe each responsibility in detail.

Table 1: Auto Trading Community Role Descriptions—Membership Applicant

ROLE	MEMBERSHIP APPLICANT	
Role Model	Auto Trading Community	
Description	Register with AutoSuperBroker and obtain community ontology and bylaws	
Responsibilities		Collaborators
	To send membership application message To send leaving community message	Destination Role: AutoSuperbroker
Explanations	By default , each broker is responsible for registering as a member in order to participant the any trading	

Table 2: Auto Trading Community Role Descriptions— Membership Inquirer

ROLE	MEMBERSHIP INQUIRER	
Role Model	Auto Trading Community	
Description	Query with AutoSuperBroker	
Responsibilities		Collaborators
	To query and receive member information of a named broker or a list of brokers	Destination Role: AutoSuperbroker
Explanations	By default , each broker is capable of querying and notifying	

Table 3: Auto Trading Community Role Descriptions— Services Advertiser

ROLE	SERVICES ADVERTISER	
Role Model	Auto Trading Community	
Description	Advertise services the broker represents to AutoSuperbroker or direct to Brokers	
Responsibilities		Collaborators
	To send advertisements or unadvertisement message of agent capabilities and services it represents	Destination Role: AutoSuperbroker Broker
Explanations	Each broker advertises the services it represents through AutoSuperBroker or direct to multiple brokers. The business relationship usually groups a set of brokers as service partner. Brokering protocol specifies how the service advertisement are represented.	

Table 4: Auto Trading Community Role Descriptions—Services Inquirer and Subscriber

ROLE	SERVICES INQUIRER AND SUBSCRIBER	
Role Model	Auto Trading Community	
Description	Request the AutoSuperbroker or other brokers for services that could not be handled by broker itself.	
Responsibilities		Collaborators
	To query and obtain brokers' information with certain service from the AutoSuperBroker, To query and negotiate contract specifications with particular broker(s), To subscribe to and obtain certain types of services	Destination Role : AutoSuperbroker or Broker
Explanations	Each broker requests services information from AutoSuperBroker and conducts business contracting with one or multiple brokers. Each broker provides the services and respond any inquirer and subscriptions.	

Table 5: Auto Trading Community Role Descriptions— Broker

ROLE	BROKER	
Role Model	Auto Trading Community	
Description	Provide service brokering in particular brokered agent system, include process member applications and unregistrations, maintain service advertises of its members and provide capabilities matching and routing. The major goal is to hide all the multi-brokering interactions from member agents.	
Responsibilities		Collaborators
	To maintain directory of member and their abilities, To receive and process service advertisements and queries, To serve the request of agents without their direct involving the contracting. The interaction details among multiple brokers are hidden from the answers to agents	Source Role: Agent
	To negotiate service contracts with other brokers.	Source Role: Broker

The basic objective of the auto trading community is to model the process chain by which businesses add value to all participants and support pair-wise contracts among multiple brokers to fulfill transactions.

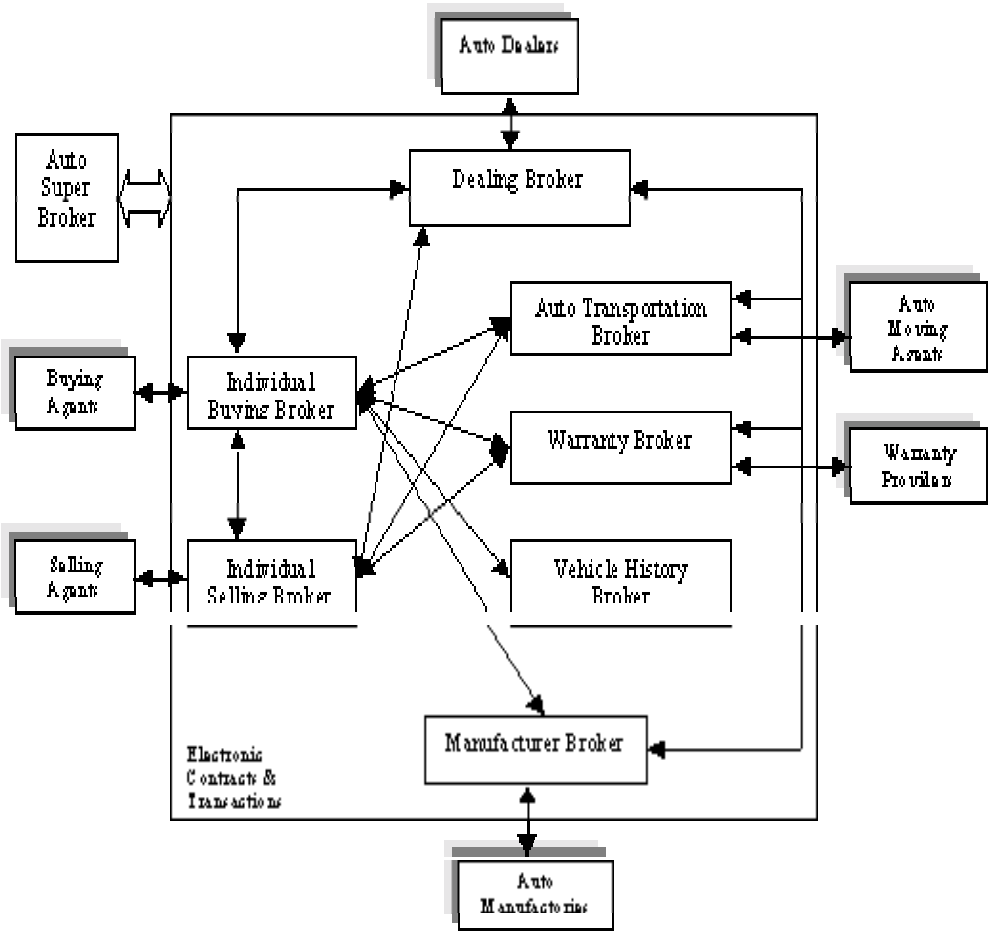


Figure 6. Interaction, Electronic Contract and Transaction in Auto Trading Community

Figure 6 illustrates the relationships between organizational units, processes in the trading community. It also illustrates the information flow between these entities.

Being members of brokered systems, individual agents can take advantage of all the capabilities and services in the whole community. However, individual agents do not have to understand all the business process and contract specifications. All business deals can be made through the representative brokers. A buying agent registers with the Individual Buying Broker without any further knowledge about dealers, manufactories, moving companies, etc. The Individual Buying Broker can buy from the Dealing Broker, the Manufacturer Broker, or the Individual Selling Broker, each of whom represents several specialized agents. In the best interest of their member agents, Brokers interact and negotiate with each other based on the service knowledge they share. A service contract usually involves specifications of multiparty. For instance, the Individual Buying Broker buys a new Honda Civic from the dealer in Miami, while it makes a deal with the auto moving companies to transport the car from Miami to Gainesville at particular date with insurance. The car is actually routed directly from the factory. Thus the additional shipping expenses are handled by the dealer. The car comes with a manufacturer warranty and 6 year extended warranty with discount price from a warranty provider, the business partner of the car dealer. This particular buying process involves various brokers and the service contracts among them.

4. Conclusion

A self-organizing community requires basic ontologies and protocols to support communications and interoperations among heterogeneous agents. The social nature of knowledge sharing carries high complexity. The capability advertisement and knowledge discovery should be achieved by message interaction among dynamic processes.

This paper addresses these issues. It designs a layered agent community architecture, defines protocols guiding interoperations across the open network, and provides a set of community-ready and application independent Java classes for e-Business application in various domains and sub-domains.

We have contributed four community protocols that guide the conversations and interoperations among different role players in the community. The knowledge discovery protocol and capability advertising protocol, where service-centric brokers share capability information with each other, are shown to enhance individual brokers' capability of solving problems within a set of agents. The joining protocol and leaving protocol, where both the superbroker and the applicant brokers evaluate the knowledge relevance, resolve the self-organizing issues in a dynamic e-Business environment.

We have also contributed a modeling that facilitates the conceptualization of the e-Business community through the encapsulation of natural business components into agents. The use of the auto trading business domain in the case study verifies the system features such as domain independence, user-friendly interface, and service and capability expandability.

Knowledge or service relevancy is one basis of self-organization of community brokers. Brokers join or leave a community based on their relevancy and consequently, their added value to the community. Currently, we use simple methods to determine such relevancy. More sophisticated algorithms need to be developed to more accurately predict a broker's relevancy to a community, as well as to bring the approach a step further towards introduction to real life business procedure and service contracting.

REFERENCES

Aglets, http://www.trl.ibm.co.jp/aglets/, Sept 1999

Carver, N. & Lesser, V., "The Evolution of Blackboard Control Architectures," *CMPSCI Technical Report 92-71*, Oct 1992.

Dan, A. et. al., "The Coyote Project: Framework for Multi-party E-Commerce," *Lecture Notes in Computer Science*, Vol.1513, pp873, Springer-Verlag, Heidelberg 1998.

Finin, T., Labrou, Y. & Mayfield, J., "KQML as an agent communication anguage," In J.M. Bradshaw, editor, *Software Agents*, AAAI Press, 1997.

Huhns, M.N. & Singh, M.P., "Readings in Agents," Morgan Kaufmann Publishers, San Francisco, CA, 1998.

JKQML, http://www.alphaworks.ibm.com/formula/jkqml, May 1999.

JATLite, http://java.stanford.edu/, May 1999.

Kendall, E.A., "Agent Roles and Role Models," http://www.labs.bt.com/projects/ibsr/papers/patterns/iaipm.ps.gz, Aug 1999

Martin, G.L., Unruh, A., & Urban, S.D., " An Agent Infrastructure for Knowledge Discovery and Event Detection," http://www.mcc.com/projects/infosleuth/publications/, Oct 1999.

Nodine, M.H., Bohrer, W., & Ngu, A., "Semantic Brokering over Dynamic Heterogeneous Data Source in InfoSleuth," *ICDE 1999*: 358-365, 1999.

Ouzounis, V., and GMD-Fokus, "R&D for New Methods of Work and Electronic Commerce," *State-of-the Art and Visions Workshops, Dec 1997-Apr 1998*, Brussels, July 1998, http://www.ispo.cec.be/ecommerce, Oct 1999.

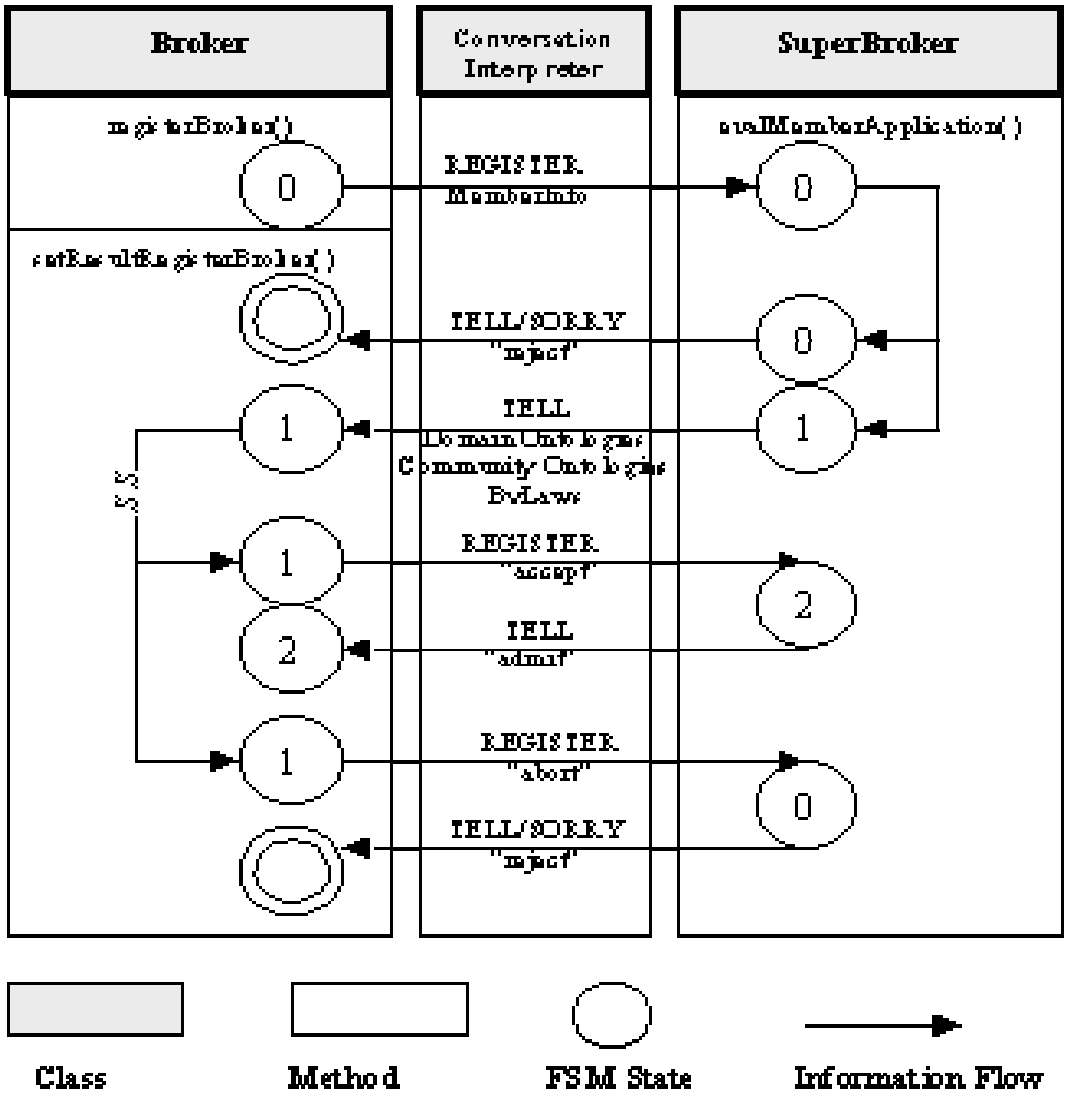
Vreeswijk, G.A.W., "Open Protocol in Multi-Agent Systems," *Technical Report CS 95-*, University of Limburg, Jan 1995.

Wang, M., "Service-Centric Brokering in Dynamic e-Business Agent Communities," Master Thesis, CISE Department, University of Florida, http://www.cise.ufl.edu/~helal/~student/thesis/Mei, Dec 1999.

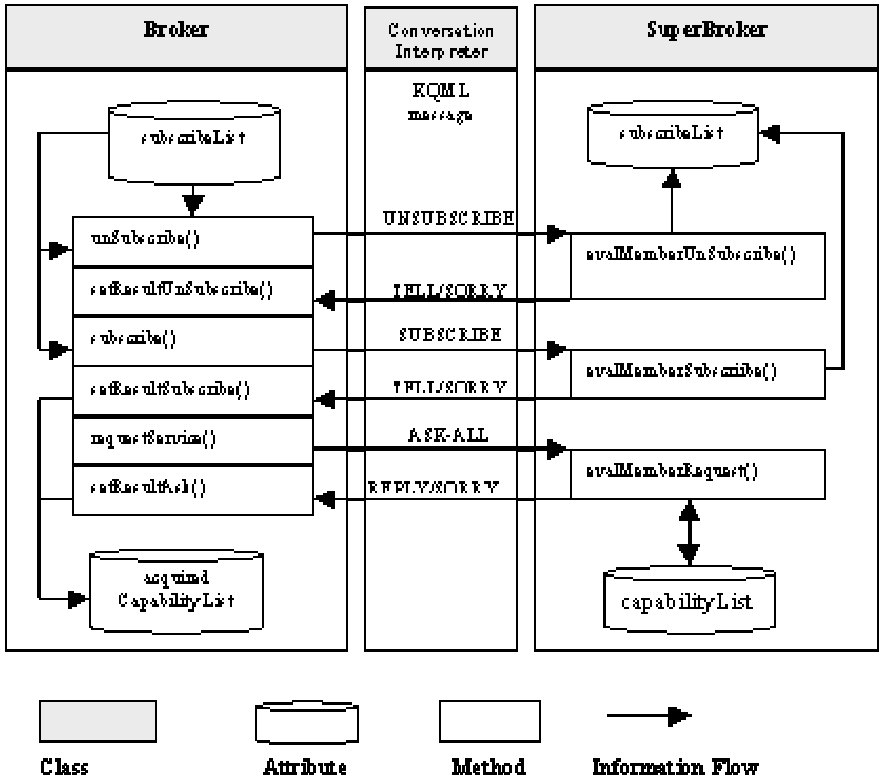
Zimmermann, H. D., "Business Media: A new approach to overcome current problems of Electronic Commerce," *Benbasat, Izak; Hoadley, Ellen: Proceedings of the 1998 Americas Conference on Information Systems AIS '98.* Baltimore, Maryland, Aug 1998.

APPENDIX A

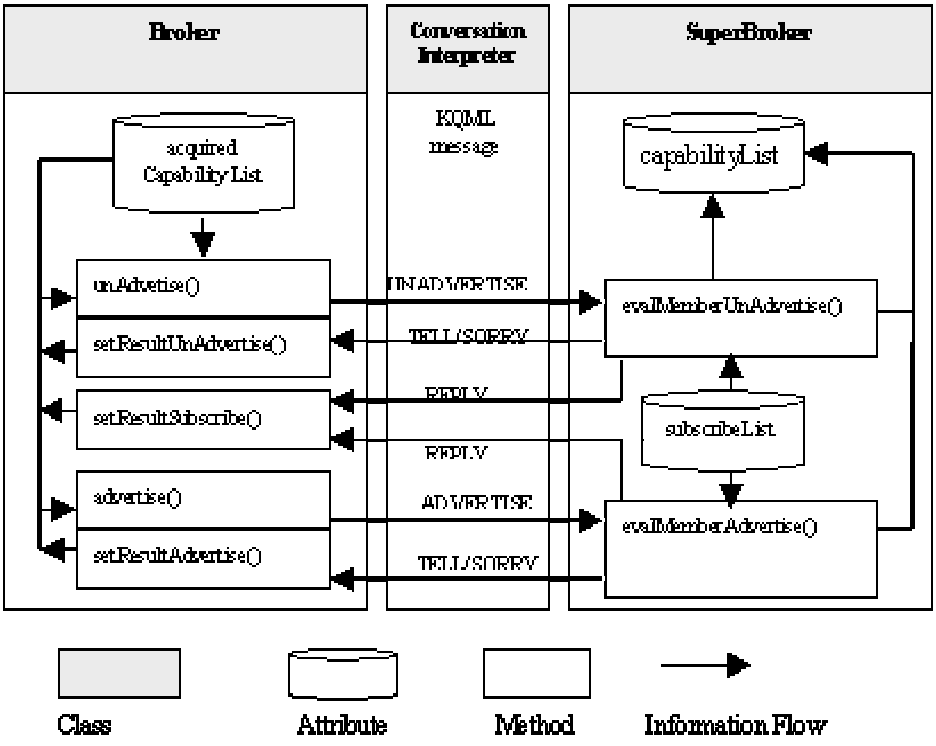
A-1. Summary of the Joining Protocol API



A-2. Summary of the Knowledge Discovery Protocol API



A-3. Summary of the Capability Advertisement Protocol API



A-4. Summary of the Departure Protocol API