Journal of Intelligent Information Systems, Vol.9, No.2, November, 2003 (pp.155~169)

A Design of Multi-Agent Framework to Develop Negotiation Systems

Hyung Rim Choi[a], Hyun Soo Kim[b], Soon Goo Hong[c], Young Jae Park[d], Yong Sung Park[e], Moo Hong Kang[f]

Department of MIS, Dong-A University

([a] *hrchoi@daunet.donga.ac.kr*, [b] *hskim@daunet.donga.ac.kr*, [c] *shong@daunet.donga.ac.kr*, [d] *yjpark@daunet.donga.ac.kr*, [e] *ys1126@daunet.donga.ac.kr*, [f] *mongy@daunet.donga.ac.kr*)

A multi-agent technology has emerged as new paradigms that can flexibly and promptly cope with various environmental changes and complex problems. Accordingly, many studies have been carried out to establish multi-agent systems in an effort to solve dynamic problems in many fields. However, most previous research on the multi-agent frameworks aimed at, on the behalf of a user, exchanging and sharing information among agents, reusing agents, and suggesting job cooperation in order to integrate and assimilate heterogeneous agents. That is, their frameworks mainly focused on the basic functions of general multi-agents. Therefore, they are not suitable to the development of the proper system for a specific field such as a negotiation. The goal of this research is to design a multi-agent framework for the negotiation system that supports the evaluation of the negotiation messages, management of the negotiation messages, and message exchanges among the negotiation agents.

Key words: Negotiation; Multi-Agent Framework

Received: October 2003 Accepted: November 2003 Corresponding Author: Hyung Rim Choi

1. Introduction

The rapid spread of computers and Internet has been quickly replacing existing offline transactions with online transactions. However, most of online transactions are possible only on the basis of a price factor. That is, online transactions are available for the limited applications such as auction or transactions of a fixed price. Accordingly, the negotiation of the products that have multiple attributes besides a price is still difficult to be done on an online basis.

Because the negotiation trade usually has multiple attributes to be considered, and also negotiation messages can be easily changed according to the inclination of counterparts, the development of negotiation system on an online basis is not an easy job. Nevertheless, the

necessity of negotiation systems is well shown in the research of Media Laboratory of MIT. According to the CBB Model (Consumer Buying Behavior Model) of the e-commerce revealed in this research, a buyer underwent six stages to buy goods: (1) the identification of the need, (2) the product brokering, (3) the merchant brokering, (4) the negotiation, (5) the purchase and delivery, (6) the service evaluation.

Accordingly, to support the negotiation, multiple attributes of a negotiation should be considered and also the inclination of trade counterpart should be analyzed. The agent system supporting a negotiation should be able to perform these functions.

An agent can be defined as a computer program that can do something on the behalf of someone else. To go into details, an agent reasons based on both perception from environment and knowledge from inside, acts, affects environment, and then continues to communicate with other agents including a user (Stuart and Peter, 1995).

The multi-agent technology to quickly and flexibly cope with the interaction of agents, diverse environmental changes and complicated problems is emerging as a new paradigm. In many fields the development of multi-agent system is in progress, and also many multi-agent frameworks facilitating the development of a multi-agent system have been suggested. These multi-agent frameworks provide a developer with API (Application Programming Interface) so that the developer may develop and control the agents within the multi-agent system, and also continue to solve problems through feedback to the developer. However, these frameworks are supporting only the general function of managing and communicating with agents, not supporting the specific function of a certain field such as a negotiation. The negotiation system needs an additional function considering the many attributes of a negotiation. Until now existing multi-agent frameworks have not provided the function to support a negotiation. Therefore, in this paper the multi-agent framework to facilitate the development of a negotiation system is designed once analyzed the attributes of a negotiation.

This paper is organized as follows. The second chapter analyzed the attributes of a negotiation and the functions of a negotiation system. The third chapter checked whether existing multi-agent frameworks could be able to support these functions. The fourth chapter, based on existing multi-agent frameworks, suggested an MAFNS (Multi-Agent Framework for Negotiation System) that can support negotiations. Finally, the fifth chapter draws conclusions followed by the contributions and further research.

2. Negotiation System

A negotiation is a way of making a decision in an effort to reach a common purpose between participants (Rosenschein and Zlotkin, 1994). In the object management group, economics, and in particular game theory, this interaction between participants is called a "protocol" or "strategy"(Guttman and Maes, 1998). Protocol is a rule of game, and means trade mechanism such as auction, reverse auction and bidding. Strategy refers to the participants' behavior to maximize their utility. In summary, a negotiation is the interaction among participants, who make a decision within the rule of a certain game, to maximize their utility. The diverse trade models that are used in the online transactions can be explained as negotiation patterns.

A lot of studies have been conducted to support diverse negotiations through a system and these studies can be divided into the two groups. The first research group ultimately aims at complete automation of a negotiation. The automated negotiation here means that a single computer or connected computers are to perform the function of negotiation without the intervention of human beings. In reality, the face-to-face negotiations are very complicated. In other hands, the automated negotiation agents in the existing research haven't included complicated processes that are necessary in the face-to-face negotiations (Beam and Segev, 1997).

Maes (1998) stressed that as one of the features of connected intelligent agents, each agent seems to be simple, but overall environments of agents are quite complicated, while acting intelligently. "Kasbah" gives a good example of an automated negotiation agent, although it is not an agent connected to network. Kasbah is a center-oriented e-marketplace to support buying and selling goods through intelligent agents. It includes an automated negotiation agent with a single attribute that buyers use the price markup strategy and sellers adopt the price cut strategy. Meanwhile, some scholars have employed a machine learning for an automated negotiation. Oliver (1996) has introduced a machine learning using genetic algorithm in an effort to teach agents an effective way of negotiating.

The second research group focuses on the Negotiation Support System (NSS) supporting the negotiation process instead of automation. The NSS provides information needed for making a decision during the negotiation or provides diverse conversation channels electronically. Different from an automated negotiation agent, a NSS depends on a human being in that it inputs limited conditions, sets problems initially and makes final decisions.

In this research, based on the analysis of negotiation systems and Sandholm's e-Medi-

used in e-auction system that should support such negotiation functions as price strategy and message evaluation, we design a framework to support a development of negotiation systems. Paula et al.(2001) pointed out that the negotiations taking place at the real world have the following features.

Table 1. Attributes of Negotiation and Functions of Negotiation System

Attributes	Explanation	Functions of Negotiation System
Multiple Attributes Negotiation	Most negotiations in the real world contain multiple attributes. For example, they contain the matters of price, delivery date, quality requirements, tax, etc.	Divide the negotiation items through parsing the messages of KQML or XML.
Similar Product Suggestion	Buyers sometimes cannot select the exact product, and so they point out the product class. In this case, sellers (or suppliers) usually recommend a similar product or an alternative product.	Definition of product category
Related Product Suggestion	When a buyer purchases a specific product, the seller can recommend related products. (for example, some products are discounted or complemented when other products are bought together.	Definition of discount and complementary relationship between products
Ultimatum	A negotiation participant can tell the counterpart that the current suggestion is a final one. A final suggestion means that if it is rejected, the negotiation will discontinue.	Provide the type of negotiation message (start, count, and finish)
Negotiation Cost	If a buyer gives up the current transaction and tries to seek a new partner, it will cause additional costs for searching. Therefore, to save additional costs, he can buy the current product.	This is not a matter of function.
Learning	The experiences in the past negotiations will be reflected in the coming negotiations.	Management and analysis of negotiation messages

As shown in the table 1, a negotiation has 6 attributes, and accordingly a negotiation system should be able to support these attributes. However, existing multi-agent frameworks don't cover all these attributes. Therefore, the functions that don't be supported by the existing multi-agent frameworks can be realized by task agent. Otherwise, a new system or components should be developed. In other words, if a developer tries to use existing multi-agent frameworks, additional functions for negotiation system should be developed as well.

3. Existing Multi-Agent Frameworks

The frameworks to develop multi-agent systems are various. Those are JAFMASS (A Java-based Agent Framework for Multi-Agent Systems), JADE (Java Agent Development Framework), FIPA-OS, DECAF, and Agent Tool. This framework designed in this paper is based on the JAFMAS and FIPA-OS Agent Platform. These two multi-agent frameworks provide the function that can realize and manage inside agents so that it can facilitate the development of multi-agent systems. Moreover, a lot of researches are actively in progress based on these two frameworks.

JAFMAS

JAFMAS developed at the University of Cincinnati uses a Java language, and provides class sets to develop the agents. A total of 16 Java classes support the realization of its system functions shown in the table 2. The development of JAFMAS has five stages.

Table 2. Analysis of JAFMAS Function

Function	Detailed Function	Explanation
Communication	Communication Protocol Layer	Support communications between agents or groups.
	Linguistic Layer (Communication Language)	Support the composition of communication messages and searching.
Social Model	Gathering Required Resources	Gather the information on the agents' attributes inside the system.
	Conversations	Support the conversations between agents by the messages generated in the linguistic layer.
	Conversation Rules	Set the communication rule of agents.
Agent	Agent	This has such attributes as agent name, host name, and server port. This is also a component for the development of agent application.
Operator Interface	Creating Agents (CreateAgent Class)	Have a list of agents' class, attribute, and subscribe, and provide them through interface.
	Agent Operator Interface	Provide the information on the agent's attributes including machine name and port number through interface.
	Conversation Operator Interface	Provide the communication messages between agents through interface.

FIPA Agent Platform

FIPA has a basic unit of AP (Agent Platform). In order to perform a cooperative job with other agents in the same platform or different platform, each agent has to be registered at least in one platform. AP provides the service for a cooperative job to the agents belonging to a platform. AP contains many components, and many APs themselves also become the components of a larger system. ACC (Agent Communication Channel) supports the message transmission between internal AP agents, and also supports the communication between APs. ANS (Agent Name Server) provides each agent with the information on the name and address of other agents. DF (Directory Facilitator) provides each agent with the information on the services and capability of the internal AP agents.

AMS (Agent Management System) manages the life cycle of internal AP agents including the registration, deletion, temporary suspension, and revival of each agent. The external AP agent is a basic performer of separate domain. That is, it is sort of an applied agent concept.

FIPA-OS consists of such components as DF, AMS and MTS (Message Transport System). DF provides other agents with yellow page containing the newly generated information of other agents. AMS monitors the life cycle of agents, and helps other agents perform well. MTS facilitates the message exchange between agents. FIPA-OS provides the following functions.

Table 3. Analysis of FIPA-OS Agent Platform Function

Function	Explanation
Task Manager	Perform the tasks generated inside the agents, and send and receive messages.
Conversation Manager	Trace the message situation between agents, and support the contents management.
MTS (Message Transport System)	Enable the messages between agents to be sent and received through diverse protocol (RMI, SSL-RMI, corbaname, http)
JESS Agent Shell	Shell function enabling FIPA-OS based agent system to use JESS
Database Factory	Provide a simple mechanism helping connection to other database
Abstract Databinding Implementation	Support the composition of special type of message. (Support XML, Java Properties, Abstract Script Binding through Datamapping Factory)

Limitations to Development of Negotiation System

While the multi-agent frameworks aforementioned have such simple functions as the generation of agent, and supporting the conversation between agents and management of agents, they don't have the function to support negotiation attributes as shown in the table 4. As shown in the table 4, the current multi-agent frameworks have no functions to define the category and relationship between products, even though these functions are absolutely necessary as an alternative method to be suggested, if the conditions of a negotiation don't coincide. In addition, the inclination of the negotiation counterpart should be analyzed; negotiation messages should be prepared, and be used to bring a better result of the negotiation. In an effort to develop these functions for a negotiation system, this paper suggests a new multi-agent framework.

Table 4. Negotiation Support Function of Existing Frameworks

Attributes	Existing Framework's Negotiation Support Function	JAFMAS	FIPA-OS Agent Platform
Multiple Attributes Negotiation	As KQML supports the messages with multiple attributes, they have a support function.	Linguistic Layer	MTS
Similar Product Suggestion	There is no function to define the category between products having different attributes.	Function required	Function required
Correlational Product Suggestion	There is no function to define the correlation between products having different attributes.	Function required	Function required
Ultimatum	Provide the same message type through communication layer.	Conversa- tions	Conversa- tion manager
Negotiation Cost	This is not a matter of function		
Learning	There is no function to manage and analyze negotiation messages.	Function required	Function required

4. Multi-Agent Framework for Negotiation System (MAFNS)

Overview of MAFNS

First of all, MAFNS is based on both the AP structure of FIPA, which has been adopted as an international standard, and JAFMAS adopting the FIPA standard. However, MAFNS has

solved and complemented the problems of existing multi-agent frameworks in the development of a negotiation system. The current multi-agent frameworks have the following limitations. When AMS judges whether a task agent has performed its task or not, AMS depends on DF (Directory Facilitator) that contains a task agent. However, in this case, the task agent that prepares negotiation messages doesn't consider the price and inclination of the counterpart agent. Therefore, the optimum result of a negotiation cannot be expected. For example, suppose agent 1 (buyer) enters into negotiations with agent 2(seller). Generally, negotiations have three stages - searching stage, negotiating stage, and follow-up stage. Accordingly, the negotiation agent has three task agents: search-counterpart-task, send-negotiation-message-task, and confirm-negotiation-task.

According to their attributes stored in the DF, search-counter-task will search for its counterpart agent, send-negotiation-message-task will prepare messages and transmit them, and confirm-negotiation-task will finalize the negotiation, send its result to the counterpart agent, and also show it to the user of the agent. However, the attributes of the task agents alone stored in the DF cannot bring the better result of negotiations. For the solution of this problem, MAFNS provides the following functions.

First, the search-counterpart-task agent doesn't consider the expected price of the counterpart agent, and depends on the parameter registered in the DF. Because of this, the negotiations are likely to fail. But MAFNS analyzes the price of the previous negotiation, and considers the price that the counterpart agent wants. Accordingly, the negotiation will have a high possibility of success.

Second, the send-negotiation-message-task agent will bring the scope of negotiation items from DF, and then prepare messages based on this information. However, this message will be prepared without considering the trade inclination of the counterpart agent. As a result, the negotiation will fail. But MAFNS will first analyze the former contents of the negotiation, and then forecast the price that the counterpart agent wishes. And according to this consideration, it prepares its message.

Finally, the confirm-negotiation-task agent sends the results to the counterpart agent and the user when the negotiation has ended. At this time, MAFNS will keep the results of the negotiation in a certain DB and use them in the next negotiation. Moreover, to suggest a similar product or a related product needed for a successful negotiation, MAFNS provides the function to define the category and relationship of the negotiation items. By using this function, MAFNS will be able to increase the possibility of negotiations.

Structure of MAFNS

Components

The major components of the multi-agent framework based on MAFNS are as follows:

- AMS (Agent Management System): manages the life cycle of agents such as the operation and suspension of agents inside the system.
- ANS (Agent Name Server): stores the information of the names and addresses of all task agents inside the system, and perform the role of mapping the logical names of task agents to the related data for real communications.
- DF (Directory Facilitator): plays the role of a "yellow page," and stores and provides the information of task agents.
- Agent Implementation: realizes an interface agent and a task agent. The interface agent provides the interface between a user and a system, and the task agent performs the job that a user wants and a common framework is provided as a super-class.
- NS (Negotiation Supporter): as a principal component of the negotiation system, it stores the messages of the negotiation, and also provides searching and inclination analysis by task agent. This also contains the information on the products and product category, and then helps defining product's relationship.

These components are based on Java language class package and its conceptual structure is shown in the figure 1.

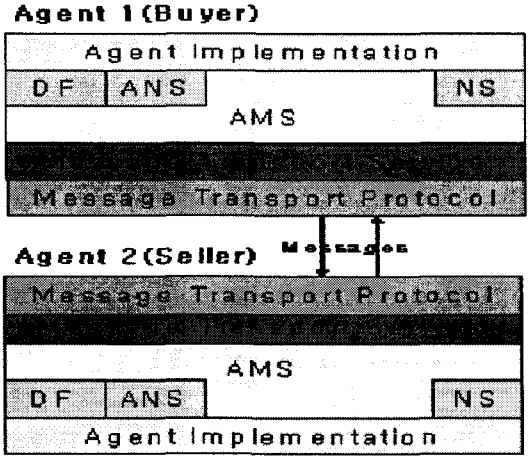

Fig. 1. Conceptual Structure of MAFNS

NS (Negotiation Supporter)

As a component to support agent implementation, NS has three functions: the function to manage the details of previous negotiations, the function to manage the results of current negotiations, and the function to define the products and product category. All the negotiation details and results are to be registered in a certain DB inside the agent to be used for the next negotiation. And these data will be used to analyze how much the negotiation items and prices of the trade counterpart agent have changed. In addition, NS provides the information on similar products and related products to increase the possibility of negotiation success.

As shown in the figure 2, the structure of NS has three classes: the negotiation-details-manager class that deals with negotiation details, the negotiation-result-manager class that manages negotiation results, and the product-manager class that supports the information on products and product category.

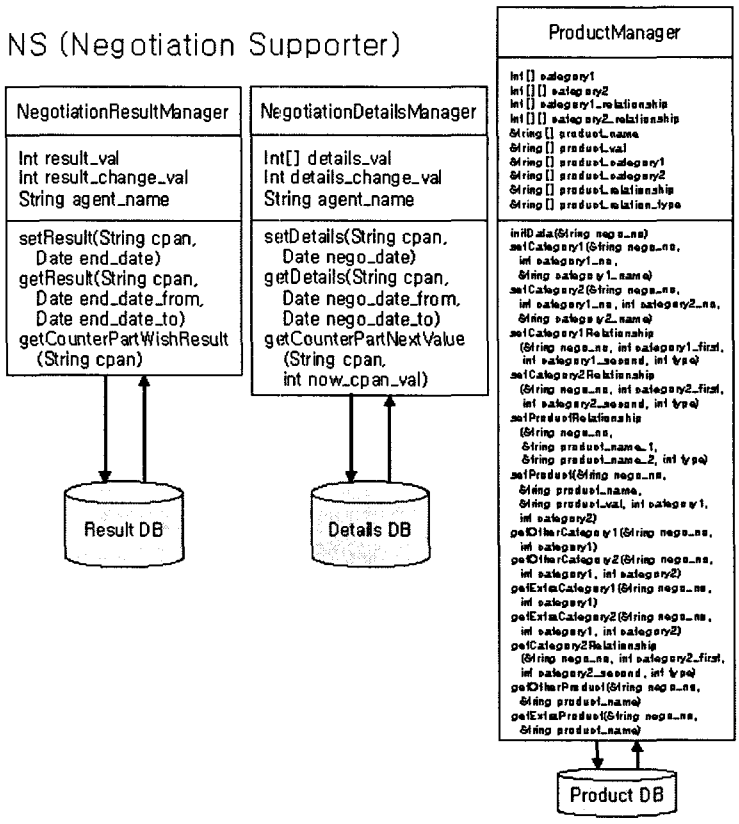

Fig. 2. Structure of Negotiation Supporter

Design of MAFNS

As mentioned above, the classes composing the agent system are divided into the two classes:the basic class in charge of the realization of agents and ACC class in charge of communications between agents. These classes are based on Java class, and the role of support methods is as follows.

AgentConnection Class

AgentConnection class performs the connection among agents. The connection will be done through such information as IP, port, and agent name. This also checks the state of connection between agents, and disconnects when the job is done.

- setAgentConnection(Address ip_adress, int port_no) : receive the ip_address and port_no of the counterpart agent, and then try to connect to a specific agent.
- getAgentConnection(String agent_name) : when setAgentConnection() succeeds in connection, the name of the counterpart agent (agent_name) will be given and connected.
- isAvailable(String agent_name) : confirm whether it is connected to the counterpart agent or not. If it is not connected, it should perform closeAgentConnection() to cut the connection to the counterpart agent.
- closeAgentConnection(String agent_name) : cut the connection to the counterpart agent (agent_name).

Conversation class

Conversation class supports the message exchange between connected agents. By using the information from the connected counterpart agent at the AgentConnection class, this can send and manage the real messages.

- getSocket(String agent_name, int port_no) : allot a socket to the connecting agent (agent_name).
- closeSocket(String agent_name) : the socket allotted to the connected agent (agent_name) will be removed.
- getMessage() : store the messages from the counterpart agent.
- sendMessage(String send_message) : send to the counterpart agent the message generated in the Messages class.

Messages class

This performs the function to generate and manage the messages for communication between agents. When generating messages, this provides KQML or XML-base messages.

- setMessageHeader(String sender, String receiver, String reply_with, String ontology, String language_type) : set the header part for message transmission.
- addMessageBuffer(String param_name, String param_val) : add the contents of message by parameter.
- getMessageString(String xml_or_kqml) : generate messages on XML or KQML-base type.
- parseMessage(String received_rnessage) : parse the messages, and store the header part and contents part by parameter.
- getParamVal(String param_name) : return the parameter prices of transmitted message.

NegotiationSupport classes

This helps a task agent support negotiations. This is composed of three classes: the class to manage negotiation details, the class to manage negotiation results, and the class to deal with the information on products and product category.

NegotiationResultManager class

- setResult(String cpan, Date end_date) : store the results of the negotiation with the counterpart agent (cpan : Counterpart Agent Name).
- getResult(String cpan, Date end_date_from, Data end_date_to) : return in rows the results of the negotiation with the counterpart agent that was held during a specific period.
- getCounterPartWishResult(String cpan) : analyze the results of the negotiation, and return the expected price that the counterpart agent wishes.

NegotiationDetailsManager

- setDetails(String cpan, Date nego_date) : stores the contents of trade with the counterparts agent.
- getDetails(String cpan, Date nego_date_from, Date nego_date_to) : return in rows the details of the negotiation with the counterpart agent during a specific period.
- getCounterPartNextValue(String cpan, int now_cpan_val) : return the expected suggestion price of the counterpart agent in response to the present price (now_cpan_val).

ProductManager

- initData(String nego_no) : bring the information on the products and product category stored in the DB by negotiation.
- setCategory1(String nego_no, int category1_no, String category1_name) : store the information of the first category by negotiation.
- setCategory2(String nego_no, int category1_no, int category2_no, String category2_name) : store the information of the second category.
- setCategory1Relationship(String nego_no, int category1_first, int category1_second, int type) : store the relationship between first categories.
- setCategory2Relationship(String nego_no, int category2_first, int category2_second, int type) : store the relationship between second categories.
- setProductRelationship(String nego_no, String product_name_1, String product_name_2, int type) : store the relationship between products.
- setProduct(String nego_no, String product_name, String product_val, int category1, int category2) : store the information on products.
- getOtherCategory1(String nego_no, int category1) : return the alternative category of the specific category. (First category)
- getOtherCategory2(String nego_no, int category1, int category2) : return the alternative category of the specific category. (Second category)
- getExtraCategory1(String nego_no, int category1) : return the category related to the specific category. (First category)
- getExtraCategory2(String nego_no, int category1, int category2) : return the category related to the specific category. (Second category)
- getCategory2Relationship(String nego_no, int category2_first, int category2_second, int type) : return the relationship between second categories.
- getOtherProduct(String nego_no, String product_name) : return the alternative product to the specific product.
- getExtraProduct(String nego_no, String product_name) : return the product related to the specific product.

The flow of functions suggested above is shown in the figure 3.

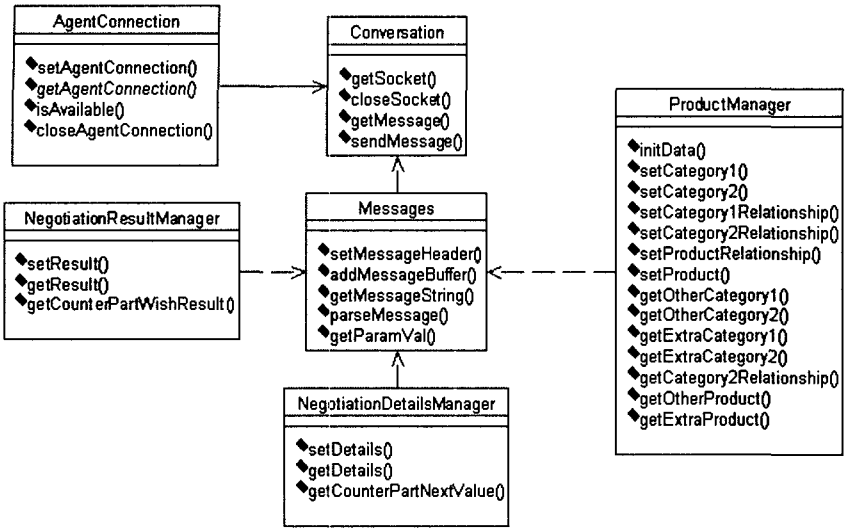

Fig. 3. MAFNS Class Structure

First, the system tries to connect to the counterpart agent through AgentConnection, and then if the connection succeeds, the negotiation starts through conversation. The negotiation messages to be used at this time are prepared by Messages. And these messages are well managed by the classes of NS. That is, the NS classes analyze the inclination of the counterpart agent, and define the attributes of products and product category to seek the better result of negotiations.

5. Conclusion

Due to the fact that the current multi-agent frameworks only provided general functions such as the generation of agent, the conversation between agents, and the management of agents, there are many limitations to the development of a specific system including a negotiation system. In this paper, we suggest a new multi-agent framework to develop a negotiation system through NS (Negotiation Supporter). NS performs the functions to suggest similar or related products, and also analyze the trade inclination of the counterpart agent, considering the attributes of negotiation.

This research has not only laid a foundation of the development of the multi-agent framework for negotiation system, but also will stimulate the development of the multi-agent framework for other specific domains. From now on, more research to develop the multi-agent

frameworks to be applied to various fields are expected to be conducted actively.

References

Beam, C., and A. Segev, *Automated Negotiations: A Survey of the State of the Art*, CMIT Working Paper 97-WP-1022, 1997.

Chavez, A., and P. Maes. "Kasbah: an agent marketplace for buying and selling goods," *Proceedings of the 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996, 75-90.

Guttman, R. H., and P. Maes, "Agent-mediated integrative negotiation for retail electronic commerce," *In P. Noriega and C. Sierra, editors, Agent Mediated Electronic Commerce, 1571, Lecture Notes in Artificial Intelligence*, 1998, 70-90.

Kang, G.. Y., J. H. Jang, and J. M. Choi, "Java-based Multi-Agent Framework," *Cognition science Society's papers*, Vol.9, No.2(1998), 25-36.

Labrou, Y., and T. Finin, *A Proposal for a New KQML Specification*, Technical Report, TR-CS-97-03, CSEE, Univ. of Maryland Baltimore County, 1997.

Oliver, J. R., "A Machine Learning Approach to Automated Negotiation and Prospects for Electronic Commerce," *Journal of Management Information Systems*, Vol.13, No.10(1996), 83-112.

Paula, E. G.., F. S. Ramos, and G. L. Ramalho, "Bilateral Negotiation Model for Agent-Mediated Electronic Commerce," *Agent-Mediated Electronic Commerce III*, Springer-Verlag, 2001, 1-14.

Rosenschein, J., and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, 1994.

Sandholm, T. W., "eMediator: A Next Generation Electronic Commerce Server," *Computational Intelligence*, Special issue on Agent Technology for Electronic Commerce, Vol.18, No.4(2002), 656-676.

Sandholm, T. W., and V. R. Lesser, "Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems," *Mellish, C. S. (ed.), Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Vol.20, No.25(1995), 694-703.

Stuart, R., and N. Peter, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

FIPA URL → http://drogo.cselt.stet.it/fipa/

JAFMAS Software Architecture URL → http://www.ececs.uc.edu/~abaker/JAFMAS/