AGENT INTERACTION MANAGEMENT AND APPLICATION IN A VIRTUAL COLLABORATIVE ENVIRONMENT

Aizhong Lin, Igor T. Hawryszkiewycz, Brian Henderson-Sellers*
* *Faculty of Information Technology, University of Technology, Sydney. POBox 123, Broadway, NSW 2007, AUSTRALIA. {alin, igorh, brian}@it.uts.edu.au.*

Abstract: The intention of managing agent interactions between agents residing in a virtual collaborative environment is to obtain some useful beliefs that can be used in agent reasoning and decision making in order to optimize further agent interactions. Agent business relationships (such as trust, loyalty, understanding and friendship) are such beliefs. This research provides an approach to the management and application of agent interaction instances. The paper firstly introduces the multi-agent system architecture built in the virtual collaborative environment. Secondly, it presents the interaction protocols designed for the software agents. Then, it describes the design and implementation of the management of interactions. Finally, it depicts a specific belief revision function for personal agents to dynamically update agent business relationships in terms of the management of agent interaction instances.

Key words: Agent, Agent Interaction, Agent Interaction Protocol, Agent Interaction Management, Agent Business Relationship, Agent Belief Revision

1. INTRODUCTION

A virtual collaborative environment called LiveNet [6] has been developed to support web-based group work. Software agents (or simply agents) built and run in the virtual collaborative environment are reusable components to manage workspace instances, goal instances, workflow instances, activity instances, groups, participants and resources. Some agents have capabilities of creating workspaces, goals, roles, participants and resources, while other agents have capabilities of gathering participants or

resources from various places (e.g., Internet or Intranet) and, finally, other agents are able to create workflow instances for specific goal instances.

When undertaking group work, we may need an agent to create a workspace instance, a goal instance and activities for the achievement of goals. Meanwhile, we may need another agent to gather participants and resources for the work. In addition, we may also need another agent to create a workflow for the work to specify its resolution. These agents all have to cooperate with each other to achieve the common goal. Their cooperation is realized by their interactions. Therefore, an important property of agents is that of interaction, leading to the notion of societies of agents.

An interaction instance occurs for a specific goal, follows a specific interaction protocol, involves a set of agents and results in a number of messages being exchanged between the agents. In LiveNet, an agent can interact with other agents for *goal delegation*, *knowledge sharing* and *cooperative group formation*. There are many interaction instances occurring among agents when they achieve common goals. Those instances require management. The benefits of managing agent interactions are:

Classifying messages based on interaction instances increases the performance of interactions

Browsing interaction history is made easier

Obtaining new beliefs for agents from agent interactions performed earlier becomes possible

The first two benefits listed above are easily understood. The third benefit introduces an application of managing agent interaction instances. The application aims to obtain specific agent beliefs - agent business relationships - from the managed agent interaction instances. Agent business relationships reflect business relationships between human users represented by the agents. The beliefs can be revised from agent interaction instances performed earlier and play important roles on further action reasoning and decision-making of the agents.

The business relationships we identify in this research are *friendship* relationships, *trust* relationships, *loyalty* relationships and *understanding* relationships. In these relationships, friendship relationships are more in-depth than other business relationships. Human users in good friendship relationships are called friends who *trust* each other, *cooperate* with each other, are *loyal* to each other and *understand* each other [11]. Polson defines "*friendships* are in-depth relationship combining trust, support, communication, loyalty, and understanding" [13]. Consequently friendship relationships combine trust relationships, loyalty relationships and understanding relationships.

Our research provides an approach to the management and application of agent interaction instances. This paper describes the approach in four major sections. The first section introduces the multi-agent system architecture

built in LiveNet; the second section describes the interaction protocols defined for software agents; the third section presents the design and implementation of the management of agent interaction instances; and the last section introduces the application of managing agent interaction instances.

2. THE MULTI-AGENT ARCHITECTURE IN LIVENET

LiveNet is built on a collaborative semantic model as shown in Figure 1. The concept "activity" in this semantic model is the implementation of a workspace. It produces well-defined outputs using many workitems, actions and interactions. A role is a collection of a group of participants. A view is a folder containing a collection of artifacts that are electronic documents produced by participants. A workflow specifies the solution for a goal in an activity. A workitem is a set of actions and interactions needed to produce intermediate outcomes that eventually produce an activity output.

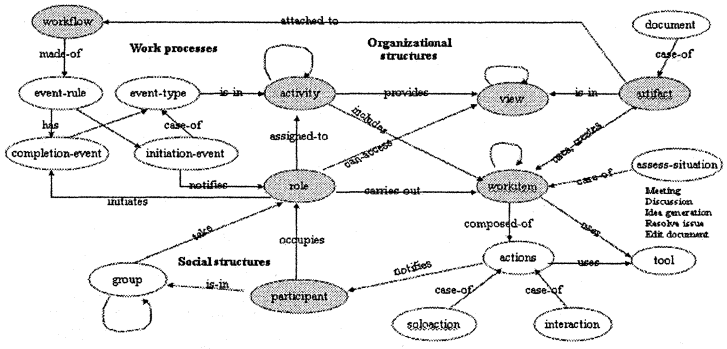

Figure 1: The collaborative semantic model of LiveNet [5]

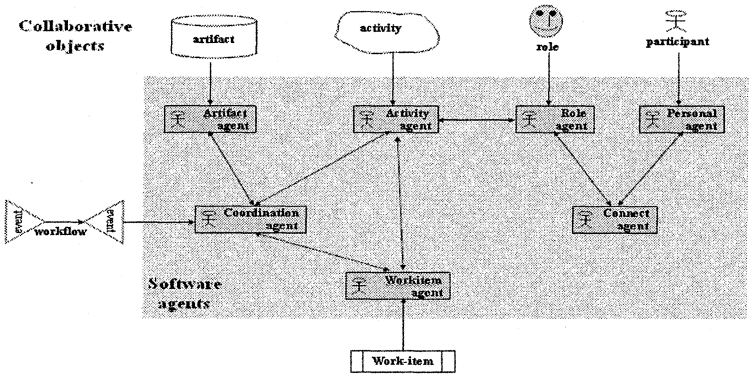

Figure 2: The multi-agent system architecture in LiveNet

The multi-agent system architecture is illustrated in Figure 2. The basic considerations for building agents in LiveNet are:

- Each participant has one and only one personal agent, which has capabilities to monitor the events related to the participants.
- Each role has one and only one role agent, which has capabilities to monitor the events related to all participants who take the role.
- Each activity has one and only one activity agent, which has capabilities to monitor events occurring in this activity.
- Each artifact may have an agent, which has capabilities to trigger a workflow instances to be started to process the artifact.
- Each workflow instance has a workflow instance monitor agent, which has capabilities to manage and monitor the workflow instance.
- Each workitem instance has one and only one workitem instance monitor agent, which has capabilities to manage and monitor the workitem instances.

The interactions between agents are classified into two dimensions. In the first dimension, considering a workspace, a personal agent could interact with a connect agent, which may interact with a role agent, to form a cooperative group for collaborative work. An artifact agent could ask a coordination agent to create or monitor a workflow instance that specifies the activities and workitem instances to produce the artifact. In the second dimension, considering different workspaces, any agent in one workspace may interact with the correspondent agent in another workspace in order to delegate a goal, share a piece of knowledge or form a cooperative group.

3. AGENT INTERACTION PROTOCOLS

Three types of agent interactions - the "delegate" type, the "share" type and the "call for joining" type - are supported by agents in LiveNet. The "delegate" type interaction is used by two agents to delegate a goal from one to the other. The "share" type interaction is used by two agents to share a piece of knowledge such as a document or a graph. The "call for joining (cfj)" type interaction is used by agents to form a cooperative working group.

In an interaction, messages are exchanged between two or more agents. An interaction instance is normally realized by a series of messages exchanged between or among agents. In LiveNet agents, messages are represented using the Agent Communication Language (ACL) [4]. Table 1 lists the performatives that are used in the three interaction protocols ("x" means that the protocol uses the performative).

Table 1: Performatives used in interaction protocols

type	performative	delegate	share	Cfj
initial	*delegate*	x		
	share		x	
	cfj			x
middle	*ask*	x	x	x
	acknowledge	x	x	x
	answer	x	x	x
	request			x
	approve		x	x
	commit	x		
	accept		x	x
	decline	x	x	x
	inform			x
terminal	*freeze*	x	x	x

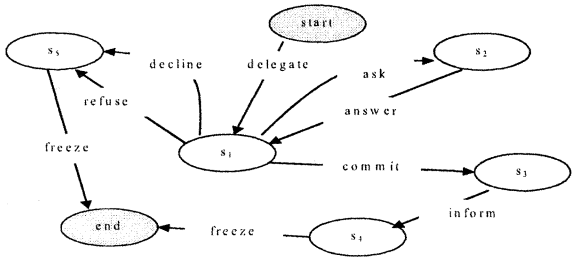

Figure 3: The "delegate" interaction protocol FSM

Interaction protocols are represented using Finite State Machines (FSMs). Figure 3 is the FSM of the delegate interaction protocol. An interaction FSM has two basic states - start and end - and one or more middle states. When an

interaction message is sent to a receiver, the interaction instance reaches a specific state. Based on the interaction protocol, the receiver agent can decide on the next message. For example, suppose using a delegate protocol, agent ag_1 sends a "delegate" message to agent ag_2, consistent with the delegate FSM (Figure 3), the current state is s_1. Agent ag_2 can choose one from four different messages (decline, refuse, commit, or ask) to reply to the "delegate" message. If agent ag_2 chooses a "commit" message, the state of Figure 3 goes to s_3. Before the FSM reaches the "end" state, agents can exchange messages according to the protocol for an interaction instance.

4. MANAGING AGENT INTERACTIONS

The management of agent interactions is modelled as a pair of components *(R, F)*, in which *R* represents the interaction instances and $F = \{f\}$ is a set of management functions.

An agent interaction instance is represented by a nine-tuple: $ii = (n, g, pr, pa, A, M, r, st, et) \in II$, in which, *II* is a set of interaction instances and:

- *n*: the name of the interaction instance
- *g*: the goal of the interaction instance
- *pr*: the protocol of the interaction
- *pa*: the patron of the interaction instance
- *A*: the set of the names of the agents involved in this interaction instance
- *M*: the set of messages exchanged in this interaction instance
- *R*: the set of results of the interaction instances
- *st*: the start time of the interaction instance
- *et*: the end time of the interaction instance

An message is represented by a eight-tuple: $m = (p, ag_s, ag_t, l, o, c, st, rt) \in M$, in which *M* is a set of messages and:

- *p*: the performative of the message
- ag_i: the sender agent of the message
- ag_j: the receiver agent of the message
- *l*: the language to represent the message
- *o*: the ontology the message uses
- *c*: the content of the message
- *st*: the send time of the message
- *rt*: the receive time of the message

A interaction result is represented by a four-tuple: $r = (gr, rl, rt, ru) \in R$, in which *R* is a set of results and:

- *gr*: the general result that indicates if the interaction goal is achieved (true) or not (false)
- *rl*: the result that indicates it is true or false that "if the patron agent ag_i of the interaction instance asks another agent ag_j to do something, the ag_j commits to do it"
- *rt*: the result that indicates it is true or false that "if agent ag_j in the interaction instance will do what ag_j commits to do for the patron agent ag_i of the interaction instance"
- *ru*: the result that indicates it is true or false that "if agent ag_j commits to do something and does it that what ag_j does is what the patron agent ag_i of the interaction instance wanted ag_j to do"

The major functions of interaction management are message generation, store, classification, retrieving and removing.

- *generate*: The "generate" function is provided in an agent to decide the next message or messages during an interaction instance. It is formalized as:

$$m_{i+1} = f_{gen}(pa, M_i)$$

where m_{i+1} is the message to be sent; *pa* is the name of the interaction protocol, its value belonging to the set *{delegate, share, cfj}*; and M_i is the set of messages that have been exchanged between agents before m_{i+1} is sent. The management function f_{gen} consists of two steps. The first step is to derive which messages it is possible to send using the finite state machine of *pa*. The second step is to decide which message from the message option to send.

- *store*: The "store" function is provided in an agent to save a message in an interaction instance or save an interaction instance to the interaction instance repository, which resides in the agent.
- *classify*: The "classify" function is provided in an agent to index interaction instances or messages in terms of given keywords. The keyword could be an interaction protocol (index interaction instances using the interaction protocol name), a patron (list the interaction instances that have this patron), an agent name (list the messages sent by this agent) and so on.
- *retrieve*: The "retrieve" function is provided in an agent to retrieve specific interaction instances or messages from the interaction repository in terms of given keywords.
- *remove*: The "remove" function is provided in an agent to delete specific interaction instances or messages from the interaction repository in terms of given keywords.

- *remove*: The "remove" function is provided in an agent to delete specific interaction instances or messages from the interaction repository in terms of given keywords.

An interaction instance is managed in the agent whose user is the patron of the instance. The interaction instances of a personal agent are listed as shown in Figure 4. An interaction record can be created, opened, and removed. When opening an interaction, all ACL messages belong to this interaction are listed. An ACL message record has an attribute to save the interaction identifier so that an ACL message belongs to an interaction instance.

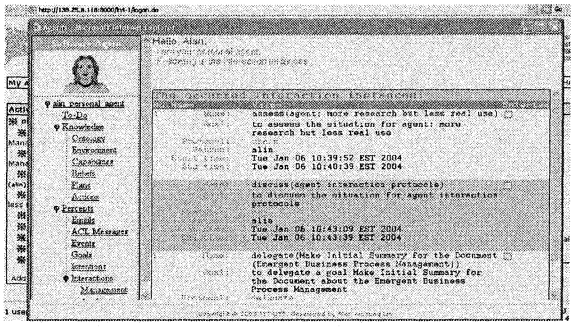

Figure 4: The list of agent interaction instances

5. AN APPLICATION OF INTERACTION MANAGEMENT

The managed agent interaction instances have many uses. Here, we consider one of its applications - to obtain agent business relationships from agent-managed agent interaction instances and apply agent business relationships to the formation of cooperative agent work team.

A LiveNet personal agent works on behalf of one and only one human user. Similarly to its human user, an agent is normally self-interested [8]. To make a group of individually self-interested agents become a "cooperative" work team, we design three strategies:

- To give opportunities to the agents who are new to the team
- To advise the agents who are not cooperative
- To encourage the agents who are cooperative

To judge an agent that is cooperative or uncooperative in a work team, the agent business relationships that can be obtained from managed agent interaction instances are employed. For example, suppose an agent ag_i is going to delegate a subprocess to another agent, the agent ag_i classifies its

- the second group contains the uncooperative agents (the agents for which ag_i believes the friendship between ag_i and those agents is less than 0.5); and
- the third group contains the cooperative agents (the agents for which ag_i believes the friendship between ag_i and those agents equals or is larger than 0.5).

After the three groups are ready, the agent ag_i will choose one agent to be responsible for the sub process in terms of the three strategies:

- firstly chooses an agent ag_j from the new agent group and delegate the sub process to ag_j;
- send a message containing an advice such as "The friendship between us is low. I think we should improve it. May I do something for you?" to all agents in the uncooperative group; and
- send a message containing an encouragement such as "You are very friendly to me. I believe we should keep this forever. I will be there when you want." to all agents in the cooperative group.

5.1 The Model of Agent Business Relationships Revision

Figure 5 is a model to harvest business relationship between agents. It is illustrated in five steps (from the bottom up, based on Figure 5):

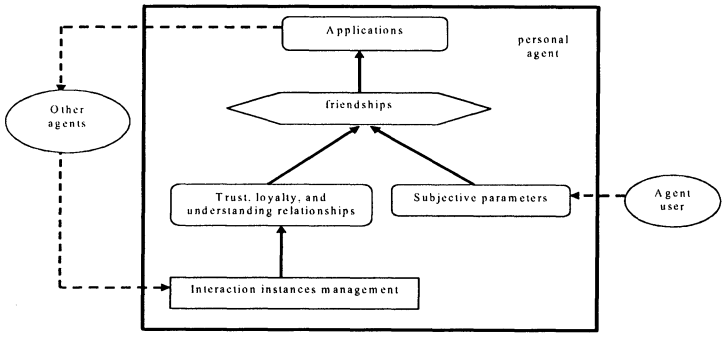

Figure 5: A model of agent business relationships revision

- During their activation, agents interact with each other by exchanging messages. An interaction message is contained in a message record and this record is contained in an interaction instance record that is saved in the knowledge base of the agent that initiates the interaction.
- After an interaction is completed, from the recorded interaction messages, the agent observes the interaction results.

- Using the interaction results, business relationships such as loyalty, trust and understanding are harvested.
- Using these business relationships and subject parameters that are set up by agent users, agent friendships are harvested.
- Finally, the business relationships are applied to assist the agent to reason and make decisions for further agent work, which returns the belief revision model back to its initial state.

5.2 The Definitions of Agent Business Relationships

As noted in Section 1, the business relationships between agents are friendship relationships, loyalty relationships, trust relationships and understanding relationships. They are defined as follows.

- *loyalty* A loyalty relationship that an agent ag_i believes exists between agent ag_i and agent ag_j in a period of time pt is the proposition that "*if* ag_i *asks* ag_j *to do something, the* ag_j *will commit to do it*". $loy_{pt}^{ag_i,ag_j}$ is the *probability* with which ag_i believes that the loyalty relationship between ag_i and ag_j is true.
- *trust* A trust relationship that an agent ag_i believes exists between agent ag_i and agent ag_j in a period of time pt is the proposition that "ag_j *will do what* ag_j *commits to do for* ag_i". $tru_{pt}^{ag_i,ag_j}$ is the *probability* with which ag_i believes that the trust relationship between ag_i and ag_j is true.
- *understanding* An understanding relationship that an agent ag_i believes exists between agent ag_i and agent ag_j in a period of time pt is the proposition that "*if* ag_j *commits to do something and does it that what* ag_j *does is what* ag_i *wanted* ag_j *to do*". $und_{pt}^{ag_i,ag_j}$ is the *probability* with which ag_i believes that the understanding relationship between ag_i and ag_j is true.
- *friendship* A friendship relationship that an agent ag_i believes it exists between agent ag_i and agent ag_j in a period of time pt is the proposition that "ag_j *is a friend of* ag_i". $fri_{pt}^{ag_i,ag_j}$ is the *probability* with which ag_i believes that the friendship relationship between ag_i and ag_j is true. A friendship relationship is the combination of the previous three business relationships.

5.3 The Method of Agent Business Relationships Revision

Typically, an agent uses the Prolog [9] logic programming language as the language of interaction message content (the Prolog language is also used to specify the beliefs of agents). Since an agent has an embedded Prolog engine for reasoning and decision-making, it can understand interaction message contents. After an interaction is completed, an assessment method, which belongs to the interact function, is activated to assess if the goal of the interaction is achieved and, meanwhile, to obtain a value for each element of R. A human process participant could also access her agent's interaction records to set or revise the value for each element of R.

The trust relationship between agents is expressed by using a real number in [0, 1]. The greater the number is, the stronger is the relationship between them. Similarly, the loyalty and understanding relationship between agents is also expressed by using a real number in the interval [0, 1]. They are calculated as:

- $tru_{pt}^{ag_i,ag_j}$ = *(the total of the rt of the interactions between* ag_i *and* ag_j *in pt) / (the number of the interactions between* ag_i *and* ag_j *in pt)*
- $loy_{pt}^{ag_i,ag_j}$ = *(the total of the rl of the interactions between* ag_i *and* ag_j *in pt) / (the number of the interactions between* ag_i *and* ag_j *in pt)*
- $und_{pt}^{ag_i,ag_j}$ = *(the total of the ru of the interactions between* ag_i *and* ag_j *in pt) / (the number of the interactions between* ag_i *and* ag_j *in pt)*

in which pt is a period of time, $pt = (pst, pet)$, where pst and pet are respectively the start time and end time of this period. The greater the pt we use, the more accurate the business relationships between agents we get. To calculate the strength of the friendship relationship from these three business relationships, an agent user has to set the weights between the friendship relationship and these three business relationships as shown in Table 2 (set up by the agent user of agent ag_{alan}).

Table 2: Examples of weights between the friendship and other three relationships

Business relationships	Weight for friendships
Trust	0.98
Loyalty	0.90
Understanding	0.7

The weights of "trust", "loyalty", and "understanding" are denoted as w_{tru}, w_{loy}, and w_{und} respectively, and the formula to calculate the friendship relationship is:

$$fri_{pt}^{ag_p,ag_q} = \left(tru_{pt}^{ag_p,ag_q} * w_{tru} + loy_{pt}^{ag_p,ag_q} * w_{loy} + und_{pt}^{ag_p,ag_q} * w_{und}\right) \Big/ \left(w_{tru} + w_{loy} + w_{und}\right)$$

6. RELATED WORK

Managing agent interaction instances can benefit group work and group members. Related work regarding agent interaction management includes that (1) the conversation layer is provided by FIPA-OS [14] to support various interaction protocols for agents; (2) the COOL language [2] was designed and implemented for agents to dynamically specify flexible interaction protocols; and (3) conversation managers [10] have been incorporated into multi-agent systems to enhance the high-level communication capability of multi-agent systems. In addition, a number of issues regarding conversation management (rather than agent conversation management) have been discussed [17] [1] [3]. However, these issues of conversation management focus on conversation analysis. The conversation layer of FIPA-OS, the COOL language and conversation managers concern more the interaction protocols than the interaction instance management.

Our examination of revising and applying agent business relationships extends previous work, particularly the following: (1) Panzarasa *et al.* [15] explore agent social relationships in an agent community by providing an agent social structure to represent the agent community and discuss the social relationships between agents in that community. (2) Hogg and Jennings [7] explore agent social attitudes (selfless, selfish, balance, social tendency and selfish tendency) that affect agent social decision-making strategies. Their social attitudes determine the agent social relationships. (3) Polson and the group in Global Friendship [13] surveyed the relationships between friendships and a set of attributes in a group of university teachers and students (200 persons) and found the most related attributes of friendship are trust (171 out of 200), honesty (113 out of 200), fun (69 out of 200), understanding (63 out of 200) and loyalty (59 out of 200). The survey tells us that friendships are closely related to trust, honesty, fun, understanding, and loyalty. (4) Marsh [12] explores the "trust" property of agents. He provides a formalization of trust, the tools necessary for trust revision and the basis for trusting artificial agents, which could form stable coalitions, take 'knowledgeable' risks, and make robust decisions in complex environments. Finally, (5) Simon [16] discussed relationships between friendliness and interaction using mathematical methods. He

concluded that "friendliness increases interaction" and "interaction increases friendliness".

7. CONCLUSION AND FUTURE WORK

Managing interaction instances benefits agents and their users in three aspects: (1) classifying messages based on interaction instances increases the performance of interactions; (2) browsing interaction history is made easier; (3) harvesting new agent beliefs from interactions performed becomes possible. This paper describes the management and application of agent interaction instances. In future, we will focus on more agent beliefs in terms of the managed interaction instances.

ACKNOWLEDGEMENTS

We wish to thank the Australian Research Council for providing funding for building reusable agents in collaborative environment. This is contribution number 04/02 of the Centre for Object Technology Applications and Research.

REFERENCES

1. Benford S., Bullock A., Cook, N., Harvey P., Ingram R. and Lee O. K., "A Model of Conversation Management in Virtual Rooms". *Proceedings of Applica '93*, Lille, France, March 1993.
2. Barbuceanu M. and Fox M. S., "COOL: A language for describing coordination in multiagent systems". *Proceedings of the First International Conference on Multi-Agent Systems* (V. Lesser, ed.), San Francisco, CA, pp. 17-24, MIT Press, 1995.
3. Finlay S. J. & Faulkner G., "Actually I Was the Star: Managing Attributions in Conversation". *Forum: Qualitative Social Research*, Volume 4, No. 1, Jan. 2003
4. FIPA, "Agent Communication Language". http://www.fipa.org/specs/fipa00003/
5. Hawryszkiewycz, I.T. (2002): "Designing Collaborative Business Systems". *Proceedings of IFIP 17th World Computer Congress, TC* Stream on Information Systems: The e-Business Challenge*, ed. Roland Traunmiller, Montreal, August 2002, Kluwer Academic Publishers, Boston, pp. 131-146.
6. Hawryszkiewycz, I.T., "Describing Work Processes in Collaborative Work". *Proceedings of The Fifth International Conference on Computer Supported Cooperative Work in Design*, IEEE Computer Society, Hong Kong, November, 2000, pp. 264-268.
7. Hogg L. M. and Jennings N. R., "Variable Sociability in Agent-Based Decision Making". *Proceedings of 6th Int. Workshop on Agent Theories Architectures and Languages (ATAL-99)*, Orlando, FL, 276-289.

8. Kraus S., "Contracting tasks in multi-agent environments". *Technical Report CS-TR 3254 UMIACS-TR-94-44*, U of Maryland, 1994.
9. Kim, S. H., "Knowledge Systems through Prolog". New York, New York: Oxford University Press, Inc.
10. Lin F., Norrie D. H., Flores, R.A., and Kremer R.C., "Incorporating Conversation Managers into Multi-agent Systems". *Proceedings of the Workshop on Agent Communication and Languages, Fourth International Conference on Autonomous Agents (Agents'2000)*. Barcelona, Spain, pp. 1-9.
11. Lewis M. and Rosenblum L.A. (eds.). Friendship and Peer Relations. N.Y. Wiley, 1975
12. Marsh P. S., Formalising Trust as a Computational Concept, *PhD Thesis*, University of Stirling, Scotland. Available http://www.iit.nrc.ca/~steve/Publications.html
13. Polson B., *http://www.cyberparent.com/friendship/what.htm*
14. Poslad S, Buckle, P and Hadingham R., "The FIPA-OS Agent Platform: Open Source for Open Standards". *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, UK, pages 355-368, 2000.
15. Panzarasa P., Norman T. J. and Jennings N. R. Modelling Sociality in a BDI framework, *Proceedings of 1st Asia-Pacific Conf. on Intelligent Agent Technology*, Hong Kong, 202-206.
16. Simon H. A.. Models of Man. *Social and Rational, Mathematical Essays on rational human behavior in a social setting*. New York: John Wiley, 1957
17. Whittaker S., Jones Q., Terveen L., "Managing long term communications: Conversation and Contact Management". *http://citeseer.nj.nec.com/472510.html*