Integrated Approach for Developing Autonomous and Interactive Software Agents

Nora Houari[1], Behrouz H. Far[2]
[1, 2] Department of Electrical and Computer Engineering, University of Calgary, Alberta, Canada
e-mail: (nhouari, far).@ucalgary.ca

Abstract

In this paper we present an approach that customize the BDI model to define a so-called "DIBRA: Desire-Intention-Belief-Rapport-Adaptation" as a generic method to support progress from individual autonomous agent concept towards interactive multiple agents. Rapport here refers to the component that connects an agent to its environment, whereas Adaptation module incorporates mechanisms of learning. The contribution of this paper is twofold; first, we propose a development approach that enables us to combine the internal and the interactive structures of multiagent system; second, the proposed methodology is applied to a real-world application targeting assistance in product development process. We believe that the five proposed tiers for multiagent systems (MAS) development serves for mastering the complexity and the difficulty of setting up effective autonomous interactive MAS.

1. Introduction

The rapid development in computing is moving towards open, dynamic and ubiquitous environments in which devices, services, and software agents are all expected to seamlessly integrate and cooperate in support of human objectives, anticipating needs, negotiation for services, acting on users' behalf and delivering services in any where, and any time.

Software Agents have been recognized as a new paradigm to build complex software systems by simplifying the complexity of distributed systems and by overcoming the limitations of the existing computing approaches. They have been emerged as a next significant breakthrough in software development and a new revolution in software. Like human agent, a software agent can carry out a task, has its own characteristics, such as autonomous, social ability, reactivity, proactiveness, cooperative, learnable and adaptable. Therefore agent is considered as a natural abstraction of the real world, in which it can model the real world with its own goals and interacts with other agents to achieve mutual benefits. In this research work we embrace an integrated approach that captures both the design of the internal (mental) structures of single agents and the interactive (social) structures that underlines multiagent interaction. Here, we seek to depict the essential building components of interactive agents that adopt in one hand the belief-desire-intention (BDI) agent model which delineates the subjective world of the individual agents and in the other hand the inter-subjective world that constitute agents interaction.

In this paper we present a generic approach that customizes the BDI model to define a so-called "DIBRA: Desire-Intention-Belief-Rapport-Adaptation". Rapport here refers to the component that connects an agent to its environment, whereas, Adaptation module incorporates mechanisms of learning.

The overall objective of the method is to be a solid foundation that serves for mastering the complexity of setting up effective multiagent systems that support progress from individual autonomous agent concept towards an interactive multiple agents.

The BDI agent model was established in the mid nineteen eighties and has become the most known and studied model of practical reasoning agents that incorporate a prominent philosophical model of human reasoning and software agent implementation [1]. The beliefs, desires, intentions are the fundamental units of an agent based system designed for dynamic complex, and unpredictable environment. Beliefs here consist of a set of assertions about the state of the world, which have an underlying propositional semantics. The general notion of assertion encompasses a full range of logical formalism [2]. Desires (or goals) are another key constituent of the system that represents some desired states. The intended plans or behaviors are intentions, which correspond to the third elementary component of the BDI model; intentions are simply executing states towards achieving the goals of the system.

Concerning the area of agent social interaction, numerous works on organizations modeling, roles and norms exist in the literature [3, 4]. In our previous work [5] we proposed a novel agent interaction model in which a clear investigation of agents' interaction scenarios are

0-7803-9402-X/05/$20.00 © 2005 IEEE

provided with the appropriate reasoning techniques, building on that and adopting an integrated methodology, we are incorporating both the internal and the social interactive structures of MAS modelling.

Our approach differs from [6] in the way the BDI agent is captured. We follow the natural style of human thought by first capturing the desires, then intentions, and beliefs. Other work [7] shares with our research the way the mental structure of the agent is identified, however it varies in two aspects. First we follow the development process starting from the high level (architecture) and going down to the design and implementation. Second and most importantly we are incorporating in our methodology key concepts of intelligent agent, instead of only the mental (BDI model), we are adding the rapport that provide the agent's interaction with its environment, and adaptation component that is the ability of learning. We are also investigating several learning mechanisms from machine learning techniques.

The application use case of the proposed approach was applied to a major product development process at one of largest enterprise dealing with collaboration across multiple companies in the area of design and development of hardware components.

The remainder of this paper is structured as follows: in Section 2 we briefly give an overview of our approach of building interactive multiagent systems. Section 3 presents the proposed DIBRA development methodology. The application of the approach within a real world case application is described in section 4, and finally conclusion and future work are given in Section 5.

2. A New Approach for Developing Intelligent Interactive Agent

Using an integrated approach of internal (mental) and interactive structure we propose a robust agent model that extends the standard BDI agent model to incorporate the interaction with the environment that allows agent to manipulate itself to coordinate with others. The agents in the system can represent individuals (human or software) or collectives, including external stakeholders such as customers, regulators or suppliers, and internal entities such as staff, departments, or systems. From the perspective of the communicative action theory and building on the three-postulated world by Habermas [8] we are modeling our interactive multiagent system from three perspectives:

-The subjective world, (how the agent perceives the world) that comprehends the feelings, beliefs, desires, experiences and intentions of the agent,

-The common social (inter-subjective) world that embodied the norms, commitments, agent relationships, and institutions to which the agents belong themselves, and which defines how agents stand towards each other, and their environment.

-The objective world of objects and states of affairs (external world) that describes how things are.

Figure 1 depicts our new approach for modeling interactive multiagent systems.

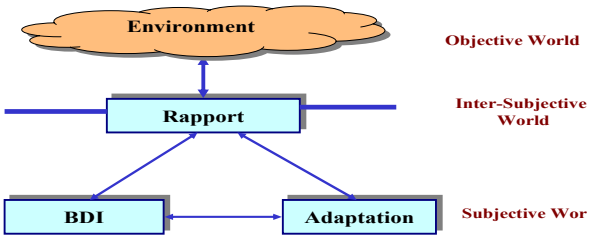


Fig. 1 The Proposed Method for modeling interactive Multiagent.

2.1 Agent Rapport Concepts

The function of the component rapport here is a mediator between the agent's internal structure and its environment to facilitate agent's interaction in terms of sensing, acting and communication tasks. In order for agent to achieve its goals or desires, it needs to interact with its environment. A more detailed look at agents' interactions (see figure 2) reveals the existing of the following two sort of interaction [9].

- Agent-Subject interaction; the subject here can be an object or another agent. A subject could be seen as any entity with attributes and state, but without having an autonomy or goal. The interaction between the agent and the subject involve sensing and acting. Sensing is the ability of the agent to receive information from the subject (e.g. receiving state and attributes of the subject). When an agent senses a subject it needs to translate the sensed data into some meaningful information for the agent. Acting is the aptitude of the agent to change the state or the attributes of the subject. When an agent acts on a subject, it has to translate the meaningful concepts of the agents into a proper change of state or attribute of the subject [9].

-Agent and another agent interaction; This is called communication, and it varies from sensing and acting, by the fact that requires a particular commitment from all parties involved in the communication with respect to the communication protocols being used, the spoken language and the vocabulary or the ontology being used [10].

Agents' communications are of special importance when agents are grouped to interact with each other to solve a problem that is beyond the capability of an individual agent. The efficiency and performance of each individual agent and the overall quality of the MAS

application depends mainly on how the agents operate and interact effectively with each other. In our previous work [5], we suggested a unified approach to capture agent-to-agent interactions, and specified the required techniques and methods to handle cooperation, coordination, and competition (see table 1)

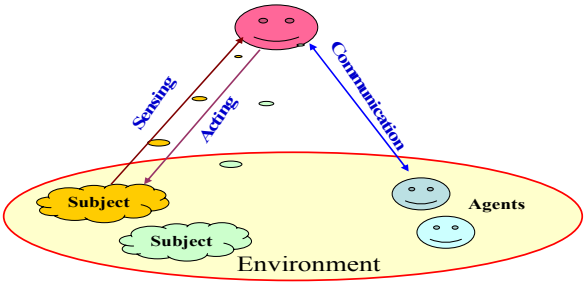


Fig. 2 Agent's Rapport with its environment: Sensing, Acting, and Communicating.

Table1. Types of Agent-to-Agent Collaboration

Interaction type	Characteristics
Cooperation	Cooperation is revealing an agent's goal and knowledge behind it to the other party. In cooperation agents have a common goal.
Coordination	Coordination is revealing an agent's goals and knowledge behind it to other party. In coordination, each individual agent has its own goal.
Loose Competition	Loose competition is revealing only an agent's goals but encapsulating the knowledge behind it to the other party.
Strict Competition	Strict competition is encapsulating both an agent's goal and the knowledge behind it to the other party.

2.2 A New Interactive MAS Development Process

In our system development process, we are using the entire software development process and following a top down structured design method, taking into account system priorities and constraints, and ensuring that the system will achieve its objectives and requirements. After requirements acquisitions, we pursue the identification of the system architecture in terms of its components and connectors that help make the system more understandable, guide development implementation of the system and evaluation of the system for future modification. Here we process an iterative and incremental modeling approach, because of the limitation of the paper length, a detailed version of the work is provided separately on the authors' web site. In this paper we present only the analysis and design phases.

Our approach follows the natural style of human thought that capture desires (goals) in the beginning, under these goals it workout the correspond intentions (plans) in order to fulfill this goals, and collect the required information and knowledge, these are beliefs. To accomplish some of the plans agents need to interact with their environment to perform some actions and/collect information this consists of rapport and at the same time they have to learn from their environment and manipulate themselves to collaborate with others, this is their ability to learn (adaptation). Figure 3 depicts our proposed approach.

In building our DIBRA model we first identify the desires from the system requirements these are the goals of the system, then capture the intentions these are the plans, followed by the required knowledge which constitutes the beliefs both explicit and implicit in the form of potential deductions based on logical reasoning; to communicate with the environment agent need a dialog (rapport) established from communication language, protocol, norms, and contracts. As an agent, it needs to adapt, that is its ability to learn from its environment; machine leaning techniques are applied for building the learning capabilities. The building blocks of our agent modeling are shown in figure 4.

3. DIBRA Agents Development Methodology

Our DIBRA development process follows the iterative and incremental software modelling phases; these are requirements, analysis, design, implementation, testing and maintenance. It has seven main steps: Problem determination and assessment, system architecture identification (components and connectors), desires (goals) identification, finding intentions (plans), designating the beliefs, specifying the rapport (interaction) with the environment, and finally building the learning ability (adaptation) (see figure 5)

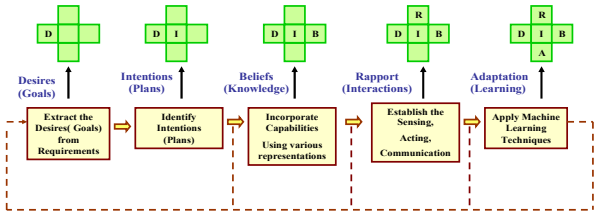


Fig.3 A New Approach for Intelligent interactive Agents: "DIBRA Model"

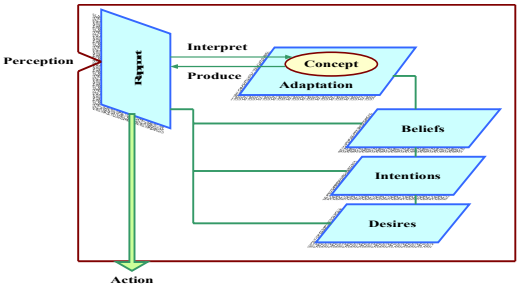


Fig. 4 Generic Building Blocks of DIBRA Agents

In this section we underline only the requirements, analysis, and design phases, we used UML for the diagrams, propositional semantics for beliefs representation, and machine learning techniques for adaptation. These phases are briefly summarized in the following steps.

Step 1: Problem Determination and Assessment: This initial step describes the problem to be solved. It is the conceptualization of the required system from the customer's point of view, and it denotes the services and functionalities that the system has to provide.

Step 2: System Architecture: In this step we identify the system components and connectors. This step is a high level description of the system that helps make the system more understandable, intellectually manageable, guide development implementation and evolution of the system for future modification.

Step 3: Capturing the Desires (Goals): The desires (goals) of the system are specified from the requirements, they are abstract artifacts. We use UML use case, collaboration, activity and sequence diagrams to represent the system services more precisely from the external point of view, we identify what services the system should provide who are the actors, what they do and for what purpose without dealing with the internal details.

Step 4: Extracting the Intentions (Plans): From the external point of view, we create internal diagrams that show how the internal entities of the system interact, these help identify intentions for the previously specified goals and reveal more details that are not obvious to extract from the external point of view. Agent's plans (intentions) refer to sets of activities to be performed and here the roles are depicted (a role includes specification of what an agent is able to accomplish it range from a particular goal to a set of goals). The roles become agents when designed the beliefs to them.

Step 5: Identifying the Beliefs (Knowledge): Beliefs are the required knowledge that the plans (intentions) use to fulfill the goals (desires). These are stored in the form of a set of assertions, which comprise standards propositional operators (conjunctions, disjunctions, negations and application) a set of quantifiers (as in predicate calculus) modal and temporal operators, and other devices for quantifying assertions with a level of uncertainty, as well as ontological assertion. After identifying basic parts of system entities: agents, desires, intentions, and beliefs, we gave names to agents.

Step 6: Specifying the Rapport (Interaction): As agents need to collaborate to fulfill the requirements, they need to interact with one another and with the external environment. We use message passing as a unique form for method invocation. Agents distinguish different type of messages as speech acts and use a collaboration protocol, this protocol is a set of public rules that dictate the conduct of an agent with other agents to achieve a desired final outcome in sharing the knowledge and performing actions that satisfy the desired goals to fulfill some utility functions. Messages are in a defined XML (eXtensible Markup Language) format and transported using the Simple Object Access Protocol (SOAP).

Step 7: Agent Adaptation: Adaptation or ability to learn is one of the properties of an intelligent autonomous agent. In our approach we are investigating several machine learning techniques build in form of algorithms, For a neural network learning algorithm for example, we develop a procedure that includes, (i) the generation of knowledge for the agent (from step5), this knowledge is expressed in the form of IF… THEN rules, then (ii) the rules are used to train the neural network. (iii) The trained neural network is used in the situations that are not covered by the rules used in (ii).

At the end of the design phase we create the agent class diagram. It contains the agent name, desires, intentions, beliefs, rapport, and a learning algorithm. At this point the system is ready to implement. It is quite straightforward, because we have almost the whole ready to code agents' capabilities and behaviours. It is possible to go back to previous artefacts of the earlier steps and make some adjustments for consistency with the interaction and the learning ability of the agents.

4. Application Scenario of the Proposed Approach

The proposed approach was applied to a real world business context that specifically relate to a major manufacturing industry domain involves in hardware component design, and production, as shown in figure 6. One of the real challenges of the manufacturing industry is how to shorten the time to go from a conceptual design to a product in a shorter period of time, also how to distribute the information on product usage to all appropriate entities, from depot maintenance, to supply

units, to repair and manufacturing components and finally to original equipment manufacturers and aligned suppliers [11]. The solution to such complex problems, without time consuming and often unavailable human intervention, can only be accomplished by invoking autonomous agents. Each agent specializes in small and distinct subsets of the overall objective of the system. Centralized control is not necessary; agents cooperate through their perception of the environment and learning ability. Their capabilities were delivered by embedding a few simple rules in each agent agenda. The rapport component of each agent helps agents to interact with each other and with a standard directory and legacy systems of the organization.

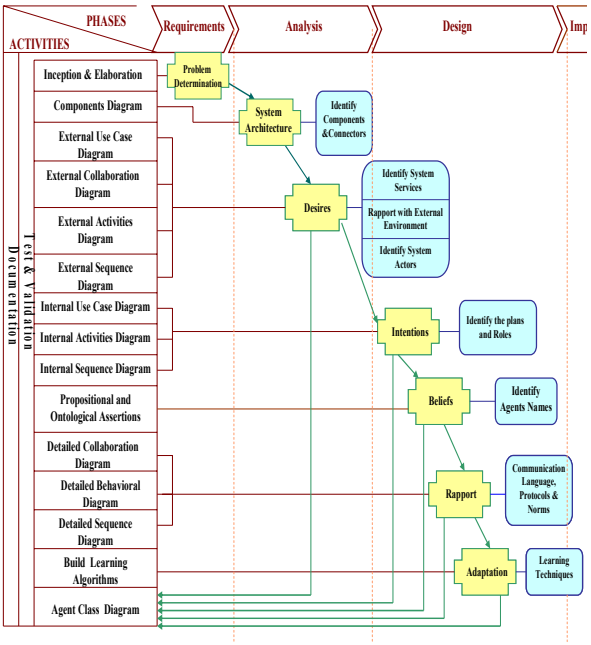


Fig. 5 DIBRA Model for Interactive MAS Agent Development Process

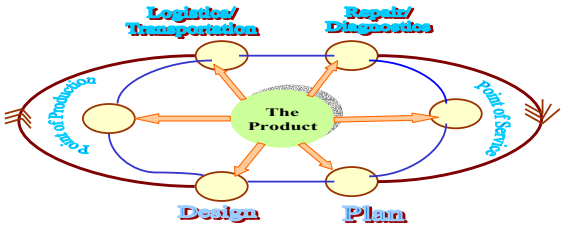


Fig. 6 Application of DIBRA Approach for Assistance in Product Development Process

4.1 Implementation

The platform chosen for the implementation of the system is Java Agent Development framework (JADE) [13]. JADE is a software development framework, fully implemented in Java, which aims at the development of multi-agent systems and applications that comply with FIPA standard. To achieve such a goal, JADE offers the following list of features:

- Distributed agent platform; the agent platform can be distributed on several hosts, each one of them executes one Java Virtual Machine.
- FIPA-Compliant agent platform; which includes the Agent Management System, the Directory Facilitator and the Agent Communication Channel.
- Efficient transport of ACL messages between agents.

We utilized the eXtensible Markup Language (XML) for agent communication, so when an agent sends message(s) to another agent (receiving agent), this message is in a defined XML format and is transported using the Simple Object Access Protocol (SOAP). The receiver agent parses the request message, processes its detail, and may return to the sender a replay also in XML format.

Agents are implemented as an extension of super class of object oriented (O.O) framework. This concrete class implements basic functionality of the software agent. The class KnowledgeRepresentation implements the data structure of the agent's knowledge [12, 13].

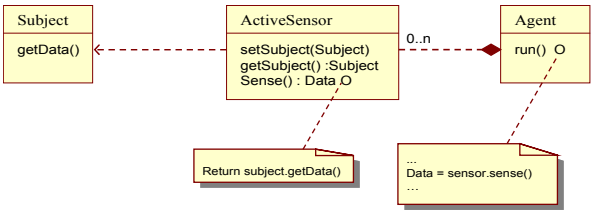
4.2 The Rapport Component Implementation

The implementation of the rapport component consists of implementing the two kinds of interaction, the agent-subject interaction, and the agent-agent communication.

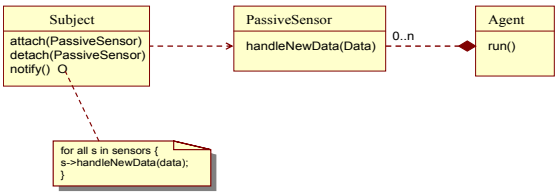
Figure 8 depicts the class diagram of the implementation of sensing. Two different types of sensing were identified; (a) active sensing: in this case the agent takes the initiative when to observe the subject (figure 6.a). (b) passive sensing; in this case the subjects takes the initiative to send data to the agent (figure 6.b). The active sensing implementation resembles to the adapter software pattern, whereas the passive sensing is similar to the observer software pattern described by [14].

In order for agents to act upon subjects in their environment, they required effectors. Unlike sensing, acting is always initiated by the agent causing modification of attributes or state of the subject. Figure 9 illustrates the class diagram of the implementation of acting. Just like the implementation of active sensing, this implementation resembles to the Adapter Software Pattern [14, 9].

The implementation of agent-to-agent interaction (communication) requires agents to be able to send and receive messages. Figure 10 outlines a class diagram of our agent communication implementation. Four different classes are required: the Agent, Communicator, Mediator, and Message class. Each agent in System has its unique AgentID, and in order to receive messages it must be register with the Mediator agent. The mediator object stores all the agents Id's. The communicator object stores the entire received message object.



(a) Class Diagram of Active Sensing



(b) Class Diagram of Passive Sensing

Fig.7 Class Diagram of Sensing (Active and Passive) Implementation

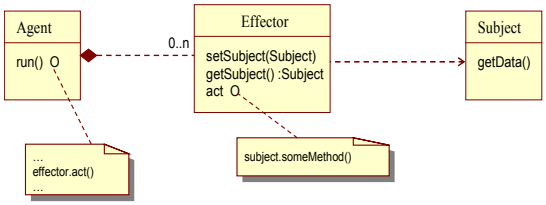


Fig.8. Class Diagram of Acting Implementation

4.3 The Adaptation Component Implementation

To implement the adaptation component we add four classes to the basic agent class represented in figure 7, these classes are: LearningAlgorithm, FunctionalityMeasure, LearningCharacteristic, and TrainingSkill. Figure 10 delineates this structure, where the control of agent's functionality is built in FunctionalityMeasure class, which is used in the LearningCharacteristic to ensure the agent is achieving its goals (desires), it enclosed the code that implements standard rules of the functionality and performance of the learning algorithm. The LearningAlgorithm which is a separate class is in charge of modifying the already existing knowledge. The training generator is configured as a separate class called TrainingSkill.

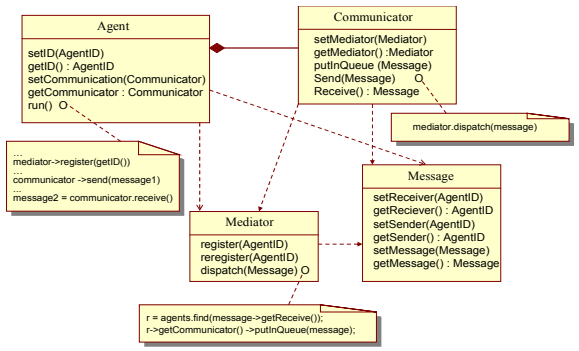


Fig.9 Class Diagram of Communication Implementation

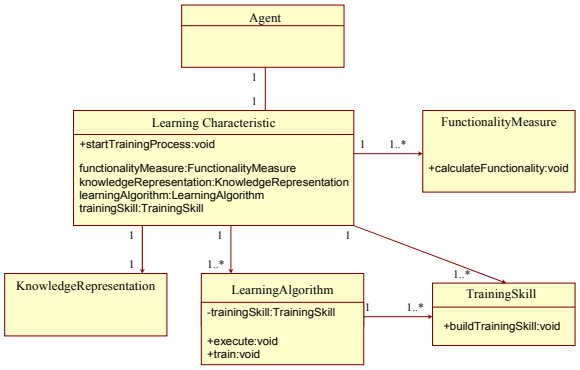


Fig.10 The Learning Class Diagram Implementation

Our experience with the system indicates that the proposed solution is a flexible and constitutes an efficient tool with enormous capabilities for effective interactive MAS applications

5. Conclusion and Future Work

This paper presents a generic development approach of MAS application that embraces both the internal (mental) structure of individual agent and interactive structure of multiagent collaboration systems. Our approach follows the communicative action theory in modeling the interactive multiagent system. We customized the BDI model to adopt the natural style of human thought that capture goals (desires) in the beginning, then workout the correspond plans (intentions), than collect the required knowledge (beliefs) to fulfill the goals. Agents are provided with interaction

capabilities to collaborate within their environment (rapport) and endowed with learning algorithms (adaptation). We use an iterative and incremental software development method that follows a top down structured design for a software system applied to the design and development of engineering products development across multiple companies. Our approach serves for mastering the complexity of building collaborative multiagent and the difficulty of setting up effective multiagent systems. Part of our ongoing work is investigating different learning algorithms. Future work will be links to other works in the area of social agent systems, especially in relation to the work of [3, 4, 2, 15].

References

[1] D. Kinny, M. Georgeff, and A. Rao, A Methodology and Modelling Technique for Systems of BDI Agents, The 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI, 61, 1996.

[2] C. B. Sidney, W. Truszkowski, "Perspectives: An Analysis of Multiple Viewpoints in Agent-Based Systems" AMKM 2003: 368-387.

[3] V.Dignum, A model for Organizational Interaction: based on Agents, founded in Logic PhD Thesis, Tekst, Proefschrift, Universiteit Utrecht, 2003.

[4] J. Filipe, Information Fields in Organization Modelling using an EDA Multi-Agent Architecture. AAAI Spring Symposium 2003 on Agent-Mediated Knowledge Management (AMKM-2003). Stanford, USA.

[5] B.H Far," A Collective View and Methodologies for Software Agents Interaction". Transactions/Magazines/Conferences: IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2004), pp. 1249-1252, May 2004.

[6] M. Einhorn, Jeffery, A BDI Agent Software Development Process, MS Thesis, (Advisor: Chang-Hyun Jo), Department of Computer Science, University of North Dakota, May 2002.

[7] C.H. Jo, C. Guobin, C. James, "A new approach to the BDI agent-based modelling" ACM Press New York, NY, USA, 2004.

[8] J. Habermas, The Theory of Communicative Action: Reason and Rationalization of Society. Polity Press. Cambridge, 1984.

[9] A.J Breemen, "Integrating Agents in Software Applications". Agent Technology Workshops. (LNAI).Springer-Verlag, 2003.

[10] M.N. Huhns, L.M Stephens: Multiagent Systems and Societies of Agents. In G.Weiss, editor, Multiagent Systems: a modern approach to distributed artificial intelligence, MIT Press (1999)

[11] National Research Council, Visionary Manufacturing Challenges for 2020, National Academies Press, 1998.

[12] J.A Sardinha, et al. 'The Agent Learning Pattern'; The 4th Latin American Conference on Patterns Language of Programming, 2004.

[13] http://jade.tilab.com/

[14] E.Gamma, R. Helm, R. Johnson, J.Vlissides : Design Patterns - Elements of Reusable Object-Oriented Software. Addison Welsey (1995).

[15] W.Vasconcelos, J.Sabater, C. Sierra and J. Querol, 2001. Skeleton-based agent development for electronic institutions. In proceedings of UKMAS.01.