

Agent-Based System Architecture and Organization

Tom Wanyama

Department of Electrical and Computer Engineering

University of Calgary

twanyama@ucalgary.ca

Abstract

While computer hardware has changed drastically in the past few years, computer software has struggled to keep pace. The centralized, monolithic programming model that was adequate when treating computers as isolated entities is poorly suited to distributed, multi-task-oriented computing. For computing to become truly ubiquitous, new distributed, multi-task-oriented programming methodologies must be developed. It is believed that distributed, multi-agent technologies offer the capabilities needed. Consequently, there are a great many projects focusing on multi-agent systems. On the other hand, at beginning, software development focused the design on algorithms and data structures. However, with increasing size and complexity of software system, the structure and the specification of entire system became more important. Software system was divided into several modules that act on each other and construct the entire properties, thus software architecture births. The architecture design for huge software system is vital to the success of entire system. Therefore, this paper reviews the different projects focusing on the architecture and organization of agent based systems. The paper first introduces the general concepts of agent-based system architecture and organization, then gives brief description of the architectures and organizations proposed by the reviewed projects

1. Introduction

The rapid development in computer technologies, and their underlying interneting capabilities, has made it impossible to continue using the centralized, monolithic programming model that was adequate when treating computers as isolated entities. Presently, interconnected computer systems are normally distributed over a wide area

leading to a distribution of competence, control, and information. Therefore, for computing to become truly ubiquitous, new distributed, multi-task-oriented programming methodologies must be developed. It is believed that distributed, multi-agent technologies offer the capabilities needed. Consequently, there are a great many projects focusing on multi-agent systems.

Agents are required to fulfill particular tasks in the real world, and these requirements have to be built into their underlying design architecture. Therefore, many agent system architectures and organizational setup for different applications have been developed as a result of research and development in the area of agent-based software engineering. Some of these are discussed in this paper.

2. Agent-Based Systems

Agent Systems especially the multi-agent systems are part of the wide research area of Distributed artificial Intelligence (DAI). DAI can be subdivided into Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS). DPS deals mainly with information management issues such as task decomposition and solution synthesis. While, MAS deals with breaking-down the problem and assigning the sub-problems to the problem solving agents with the best abilities to solve the particular sub-problem. Although each agent has its own goal and interests, the assigning of the sub-problems is done in such away as to solve the global problem in the most appropriate and efficient manner.

Agents are autonomous or semi-autonomous hardware or software systems that carry out

tasks in complex, continuously changing environments. A MAS consists of a group of agents that can take specific roles within an organizational structure (Finin et al, 1994).

2.1 Capabilities of Agents

In order to cope with the tasks for which they are designed, agents have the following basic capabilities;

- **Reactive:** That is, agents must react timely and appropriately to unplanned events and to changes in the environment.
- **Goal oriented:** Agents act in a purposeful manner.
- **Communicative:** They should be able to communicate with the environment, other agents, and/or people.
- **Adaptive:** They should be able to change their behavior due to previous experience
- **Autonomous:** They must exercise control over their own actions.

- **Temporally Continuous:** Agents must possess the ability to running continuously.

2.2 Single-Agent Systems

Single-agent systems are based on the centralized process model. In these systems, there is a single agent which makes all the decisions, leaving all the other agents to act as remote slaves. Therefore, single agent systems may have a number of entities such as transducers, actuators and/or robots. However, all entities send their perceptions to, and receive their actions from the same central processor

The environment of a single-agent system may have other agents. However, these agents act as actuators or sensors because they do not possess goals of their own.

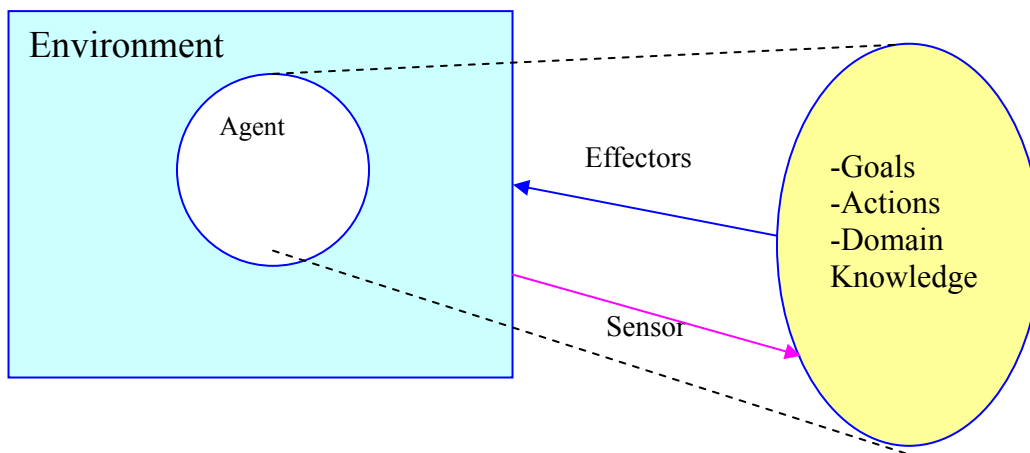


Figure 1: General Single-Agent Framework

2.3 Multi-Agent Systems

Various definitions for multi-agent systems have been proposed by different researchers and/or disciplines. However, from DAI point of view, a multi-agent system is a loosely coupled network of problem-solving agents that work together to solve problems that none of them could solve alone. The main difference between multi-agent systems and single-agent systems is that in multi-agent systems several agents exist, and they are aware of each other's goals and actions. Besides being aware of each other's intentions and behavior, in a fully general multi-agent system, agents also communicate with one another, either to

help an individual agent achieve its goal, or in a rare case, prevent it.

Multi-agent systems are composed of several autonomous entities which have the following general characteristics;

- Each agent has incomplete capabilities to solve the problem
- There is no global control
- Data is decentralized
- Computation is asynchronous

Figure 2 below shows a multi-agent system with multiple agents, some with communication capabilities and others without

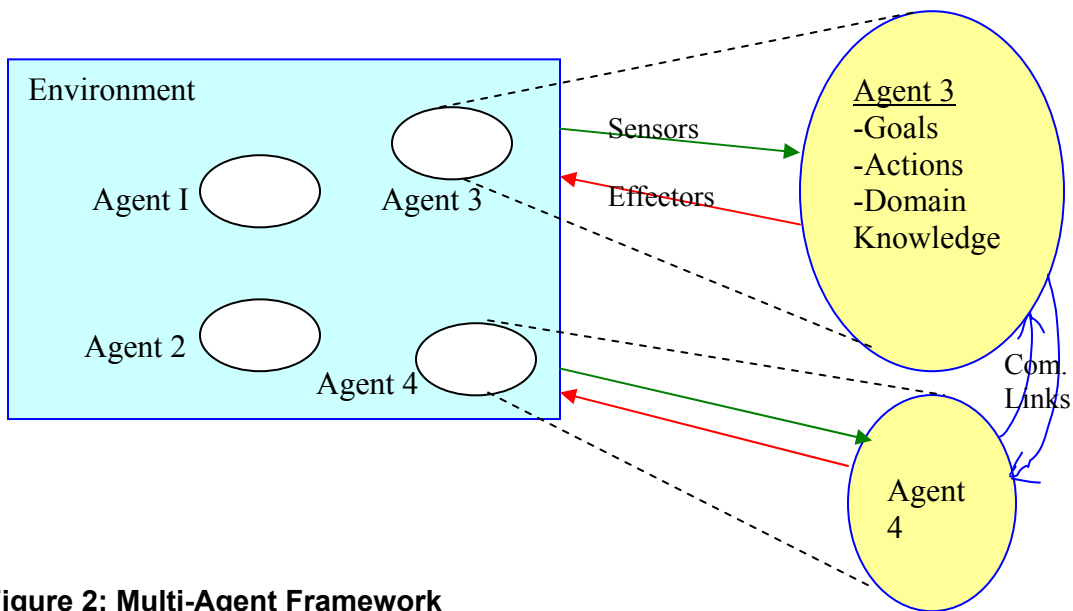


Figure 2: Multi-Agent Framework

2.3.1 MAS Knowledge Sharing

Knowledge level communication is required for MAS to share knowledge. To achieve this, a number of Agent Communication Languages (ACL) have been proposed. Among them, Knowledge Query and Manipulation Language (KQML) has been proposed as a protocol for exchange of information, and Knowledge Interface Format (KIF) has been proposed as a formal syntax for representing knowledge (Far et al, 1999).

2.3.2 MAS Problem Solving

To solve the problem, agent in MAS interact in one of the following ways, cooperation, coordination or competition.

- **Cooperation;** The agent reveals its goal and symbol structure to the other party, and both agents have the same goal.
- **Coordination;** The agent reveals its goal to the other party, but the agents have different goals.
- **Competition;** This is divided into loose competition and strict competition. In loose competition, the agent reveals its goal to the other agent, but encapsulates its symbol structure. In strict competition, the agent encapsulates both its goal and symbol structure.

2.3.3 Advantages of MAS

MAS can be used for both distributed and centralized systems. For example, multiple agents can be used to speed up systems by providing means for parallel programming.

Another benefit of multi-agent systems is their scalability. That is, a new agent can easily be added to the multi-agent system, because it is inherently modular. Generally this is more easily done than adding new capabilities into monolithic systems.

For programmers, modularity of MAS leads

to simpler programming. Instead of working with one centralized agent, programmers easily identify subtasks and assign control of these subtasks to different agent. This also solves the problem of sharing time of one centralized agent between separate tasks.

3. Internal Agent Architecture

The internal architecture of a software agent is the organization of the parts that make up the agent such that it achieves its intended goals in the outer environment. Therefore, many research organizations have proposed a number of internal architectures for agents. Far et al proposes the architecture in figure 3 below for a generalized agent.

A generalized agent is defined as an information processing entity that acts on basis of representation, using as input information the sensor receive from the external environment, perceiving, conceptualizing, interpreting, and performing action so as to achieve a desired goal (Far et al, 2000).

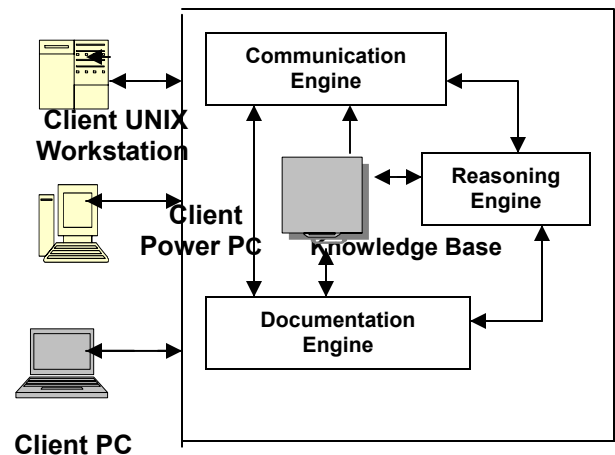


Figure 3: Internal structure of A Generalized Agent

Edmonds and Moss propose the internal architecture shown in figure 4 for an economic agent. The agent has many models of its environment. Once started the agent incrementally develops and propagates these models according to a fitness function which is based on its memory of past data and effects of its actions as well as the complexity and specificity of its models. It

then selects the best such model according to that measure. From the best such model and its goals, it attempts to determine its action using a search-based, deductive or quasi-deductive mechanism. It then takes that action and notes the effects in the environment for future use (Edmonds and Moss, 1997).

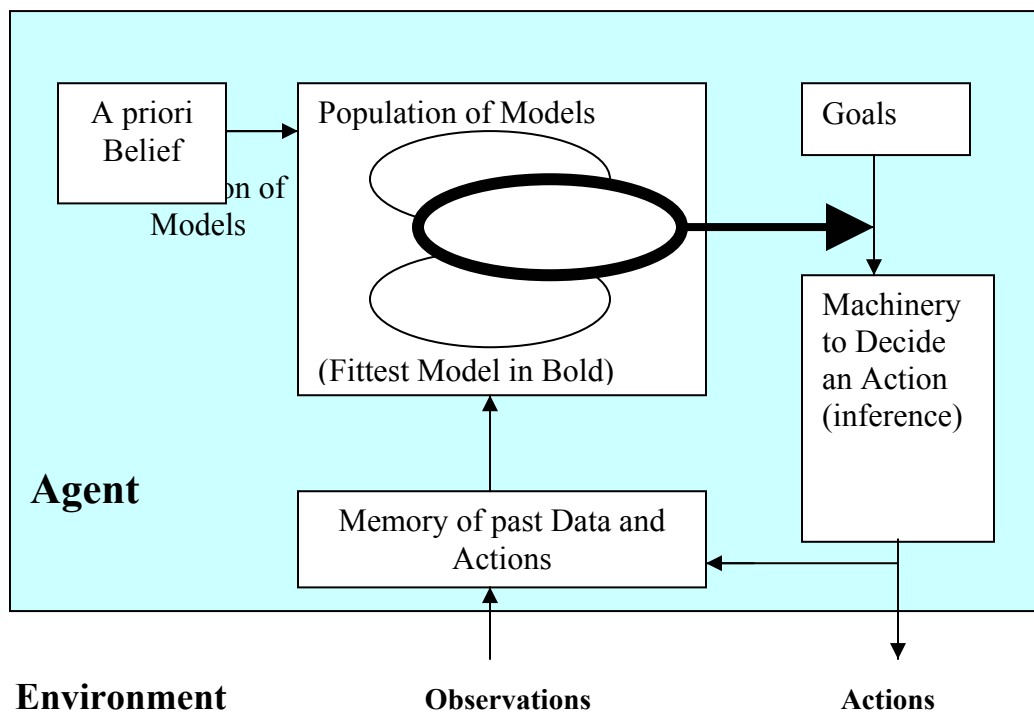


Figure 4: Basic Internal Structure of a Simplified Economic Agent

Individual agents in the New Zealand Distributed Information Systems (NZDIS) research project model, have architectural organization shown in figure 5. In this architecture, the Agent Executive is in the control of the agent. A message handler operates one or more input and output queues of the agent and makes them available to the Agent Executive. The Belief States component contains the current state of the agent in a declarative form. The agents also have separate conversation

components, each of which maintain the state of one agent's conversation dialogues in accordance with a conversation policy appropriate to the agent's role in that conversation. The component called Role comprises of a conversation policy, a set of goals to be achieved, and the plan required to fulfill the goals. Some agents in this model have additional capabilities shown as capabilities in figure 5 (Purvis et al, 2000).

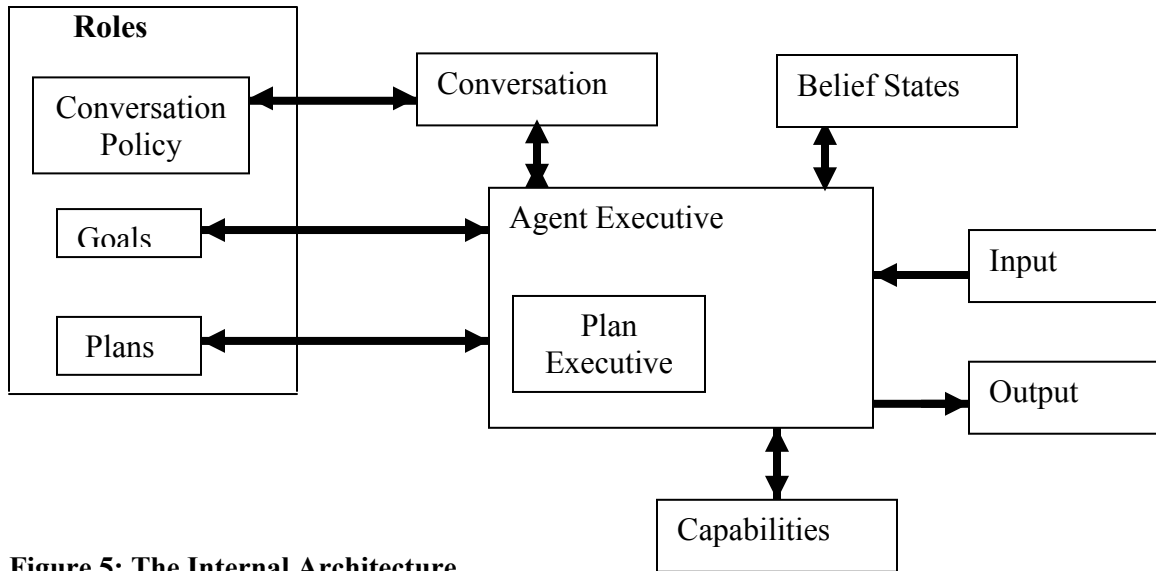


Figure 5: The Internal Architecture of the NZDIS Agent

3.1 Comparison of the Three Internal Structure Models

The three models reviewed above were designed for different purposes. However, they all have four basic component required for a piece of

software to acts as an agent. The four components namely; control, knowledge, goals and communication, are implemented differently in the different models as illustrated in table 1 below.

Table 1: Comparison of the Three Agent Architectures

Agent Component or Capability	Generalized Agent (Far et al)	Economic Agent (Edmonds & Moss)	NZDIS Agent (Purvis et al)
Agent Control	Reasoning Engine	Mechanism to Decide actions	Agent Executive
Knowledge	Knowledge Bank	Population of Models & Knowledge of Past Data and Actions	Capabilities and Plans
Goals	Reasoning Engine	Goals	Goals
Communication	Communication Engine	Actions and Observations	Conversations
Documentation	Documentation Engine	Observations	Belief State

4. Software Agent Organizations

An organization is a goal oriented coalition of software agents in which the agents are engaged in one or more tasks. Control, knowledge and capabilities are distributed among the agents (Far et al, 1999).

The structure of an organization defines the roles of the various internal actors, their responsibilities, defined in terms of tasks and goals they have to achieve, and resources they have been allocated. Secondly, an organization defines how to coordinate the activities of various components and how they depend on each other. Such dependences may involve both actors of the organization and its environment. For example, partners, competitors or clients.

Organizational structures can be grouped into two main categories, namely; centralizes and decentralized organizational structures. The centralized structure employs the master/slave coordination approach, like in the blackboard technology. The decentralized architecture uses contracting approach to coordinate activities of the involved agents.

An organization style defines a class of organizational structure, and offers a set of design parameters that can influence the division of labor and the coordination mechanisms, thereby affecting how the organization functions. Design parameters include, among others, task assignment, standardization, supervision and control (Kolp et al, 2002).

A number of agent-based software organization structures have been proposed by different individual researchers and research organizations. These include; Jini,

Generalized Agent and Open Agent Architecture.

4.1 Blackboard

In this approach of multi-agent systems, each agent has partial knowledge and capability to solve the impending problem. That is, none of the agents can solve the problem alone. Secondly, agents strictly communicate through a common blackboard whose information is accessible to all agents.

A blackboard system consists of three modules, namely, *Knowledge Source* (KS) which usually are agents, the *Blackboard* which is the global database, and the *Control Component*, also called the moderator. When a partial problem is to be solved, the control unit assigns the task to a suitable agent selected from a list of candidates that can possibly solve the problem. Figure 6 below shows the blackboard architecture.

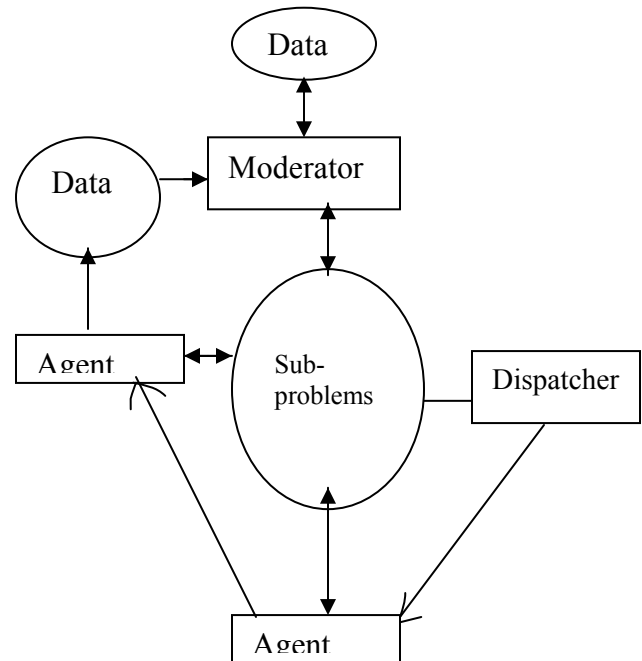


Figure 6: The Blackboard Architecture

4.2 The Open agent Architecture

The Open Agent architecture (OAA) is based on the blackboard technology. It allows individual agents to communicate by means of goals posted on a blackboard. The facilitator is a specialized agent that is responsible for coordinating agent communications and cooperative problem-solving. OAA system configuration can have more than one facilitator. Application agents are specialists that provide a collection of services of a particular type, and meta-

agents are those whose role is to assist the facilitator agent in coordinating the activities of other agents. The user interface agent connects the OAA system to the external world. In some systems, this agent is implemented as a collection of “micro-agents”, each monitoring a different modality (point-and-click, handwriting, pen gestures, speech), and collaborating to produce the best interpretation of the inputs. Figure 7 below shows the Open Agent Architecture

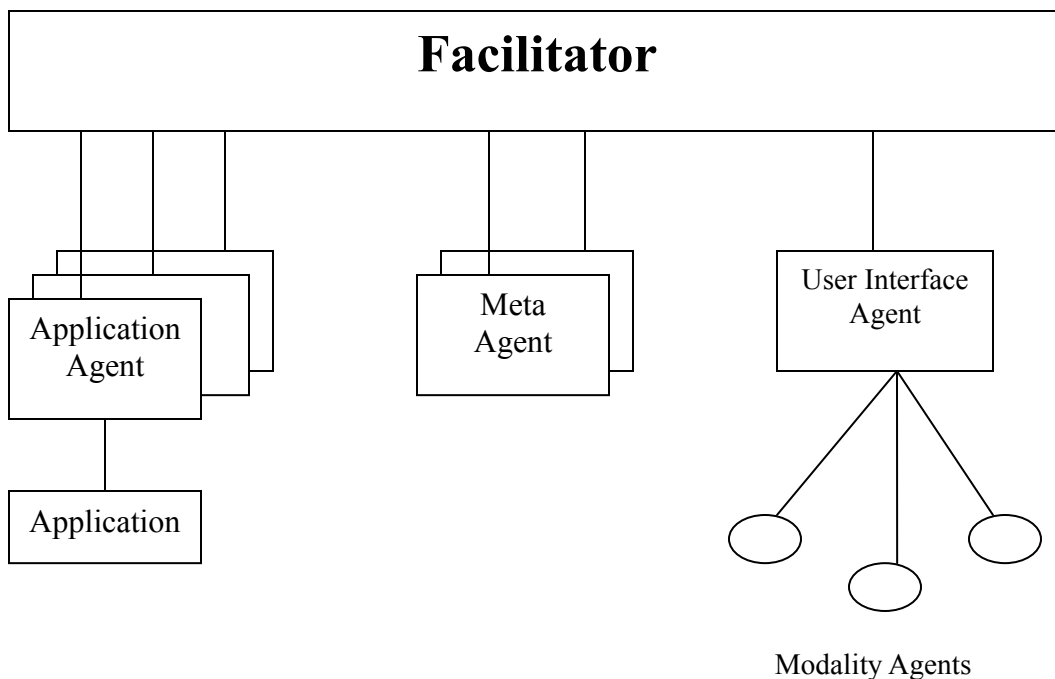


Figure 7: OAA System Structure (Martin et al)

5. Conclusion.

There are many projects focusing on multi-agent systems, architectures and organizations. As a result, many internal agent architectures and agent-based system organization architectures have been developed. Most of the architectures have

been developed to solve particular problems. However, some of them can be modified to suit other problems. Some general architectures also exist, for example the Generalized Agent and the Open Agent Architecture. Such architectures need to be studied well before adapting them to any system, because they also have some limitations.

References

1. Bilow C. S., *An Architecture for the Integration of New and Legacy Software into Broadcast Environment*, URL: <http://www.broadcastpapers.com/assets/GVGArchitectureInteg06.htm>
2. Chuang T., Yadav S. B., *An Agent-Based Architecture of an Adaptive Support System*, URL: <http://hsd.baylor.edu/ramsower/ais.ca.97/papers/chuang.htm>
3. Edmonds B., Moss S. *Modeling Bounded Rationality Using Evolutionary Techniques*, Center for Policy Modeling, Manchester Metropolitan University, 1997.
4. Far B. H., *Agent Oriented Software Engineering*.
5. Far B. H., Hajji H. Saniepour S., Soueina S. O., *Formalization of Organizational Intelligence for Multiagent System Design*, IEICE Transactions, Vol.E83-D, No. 4, 2000.
6. Finin T., Fritzson R., McKay D., McEntire R., *KQML as an Agent Communication Language*, Proceedings of the Third International Conference on Information and Knowledge Management, 1994.
7. Frankline S., Graesser A., *Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.
8. Martin D. L., Cheyer A. J., Moran D. B., *The Open Agent architecture: A framework for Building Distributed Software Systems*, Artificial Intelligence Center, SRI International, USA.
9. Padfovan B., Sackmann S., Eymann T. Pippow I., *A Prototype for an Agent-based Secure Electronic Marketplace Including Reputation Tracking Mechanisms*, Institute for Computer Science and Social Studies, Albert-Ludwing-University Freiburg, Germany.
10. Purvis M., Cranefield S., Bush G., Carter D., McKinlay B., Nowostawski M., Ward R., *The NZDIS Project: An Agent-Based Distributed Information System Architecture*, University of Otago, Dunedin, New Zealand.
11. Rajesh V., Canfield K., Quirologico S., Silva M., *An Agent-Based Architecture for Interoperability among heterogeneous medical Databases*, Department of Information Systems, University of Maryland, USA.
12. Vetter M., Pitsch S., *An Agent-Based Market Supporting Multiple Auction Protocols*, Stuttgart, Germany.
13. Yen J., Hu J., Bui T. X., *Intelligent Clearinghouse; Electronic Marketplace with Computer-mediated Negotiation Supports*.