Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

Intelligent Agents

Ira Rudowsky
Brooklyn College

Intelligent Agents

Ira Rudowsky
Brooklyn College
rudowsky@brooklyn.cuny.edu

ABSTRACT

A search on Google for the keywords "intelligent agents' will return more than 330,000 hits; "multi-agent" returns almost double that amount as well as over 5,000 citations on www.citeseer.com. What is agent technology and what has led to its enormous popularity in both the academic and commercial worlds? Agent-based system technology offers a new paradigm for designing and implementing software systems. The objective of this tutorial is to provide an overview of agents, intelligent agents and multi-agent systems, covering such areas as:

1. what an agent is, its origins and what it does,
2. how intelligence is defined for and differentiates an intelligent agent from an agent,
3. how multi-agent systems coordinate agents with competing goals to achieve a meaningful result, and
4. how an agent differs from an object of a class or an expert system.

Examples are presented of academic and commercial applications that employ agent technology. The potential pitfalls of agent development and agent usage are discussed.

Keywords

Agents, intelligent agents, multi-agent systems, artificial intelligence

WHAT IS AN AGENT?

Historical Context

Artificial Intelligence (AI) and agent systems have been closely related over the last thirty years. AI is interested in studying the components of intelligence (e.g., the ability to learn, plan) while the study of agents deals with integrating these same components. This distinction may seem to imply that all the problems within AI must be solved in order to build an agent. But, as Oren Etzioni points out, this is not the case – 'Intelligent agents are ninety-nine percent computer science and one percent AI' (Etzioni, 1996). While AI techniques may be drawn upon to build agents, not all the capabilities are required by an agent and thus not all AI problems need be solved before building an agent. For example, the ability to learn may not be a desirable trait for an agent in some situations while it is certainly a component of AI.

Between 1960 and 1990, AI witnessed a great deal of progress in many sub-areas such as knowledge representation and inference, machine learning, vision, robotics. In addition, various advancements in computer science and computing e.g., multitasking, distributed computing, communicating processes, real-time systems and communication networks made the design, implementation and deployment of agent based systems possible, at least in principle. The potential applications in distributed databases, mobile computing, information gathering, and collaborative computing that take advantage of these advances in AI and computer systems pose a strong argument for the development of intelligent agents and multi-agent systems.

But is all this in touch with the reality? One need look no further than NASA's Deep Space 1 (DS1) project where an artificial intelligence system was placed on board to plan and execute spacecraft activities. In contrast to remote control, this sophisticated set of computer programs acts as an agent of the operations team on board the spacecraft. Rather than requiring humans to do the detailed planning necessary to carry out desired tasks, Remote Agent will formulate its own plans, by combining the high level goals provided by the operations team with its detailed knowledge of both the condition of the spacecraft and how to control it. It then executes that plan, constantly monitoring its progress. If problems develop, Remote Agent in many cases will be able to fix them or work around them. If it is unable to find a fix or a work around, it can request help from its human counterparts. Remote Agent operated DS1 spacecraft during two experiments that began on May 17, 1999, when it ran the on-board computer more than 60,000,000 miles from Earth. The tests were a step toward robotic explorers of the 21st century that are less costly, more capable, and more independent from ground control. These intelligent agents can have the potential of making space exploration of the future more productive while staying within NASA's limited

budget. By transferring functions normally performed by people to a remote agent, a spacecraft may be more agile in responding to unexpected situations it encounters. In addition, by assuming responsibility for on-board tasks that currently require human intervention from Earth, Remote Agent will permit spacecraft to fulfill their mission while greatly reducing the time consuming and labor intensive communications to and from mission control. This ability will enable NASA to achieve its goal of launching many more spacecraft into the solar system while staying within budget. So we have what looks like a winning idea – what's it all about?

Defining an Agent

There is no universally accepted definition of the term agent. Russel and Norvig (1995) define an agent as an entity that can be viewed as perceiving its environment through sensors and acting upon its environment through effectors. (Coen, 1995) views software agents as programs that engage in dialogs and negotiate and coordinate the transfer of information. Wooldridge and Jennings (1995) state that an agent is a hardware and/or software-based computer system displaying the properties of autonomy, social adeptness, reactivity, and proactivity. Others (Brustolini, 1991; Franklin and Graeser, 1996; Maes, 1995; Hayes-Roth et al, 1995; Gilbert et al, 1995) offer variations on this theme. There is a consensus that autonomy, the ability to act without the intervention of humans or other systems, is a key feature of an agent. Beyond that, different attributes take on different importance based on the domain of the agent.

Figure 1 depicts a high-level view of an agent within its environment. An agent receives input from its environment and, through a repertoire of actions available to it, reacts to it in order to modify it. Generally, in domains of reasonable complexity, an agent will not have total control over its environment. Thus, the same action performed twice in seemingly identical situations might appear to have completely different outcomes. Failure is also a possibility i.e., the action taken by the agent may not produce the desired effect at all.

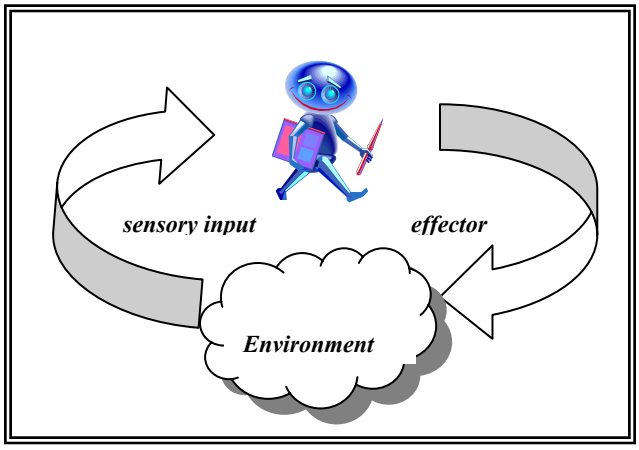

Figure 1. Agent interacting with its environment

Agent Environments

The critical decision an agent faces is determining which action to perform to best satisfy its design objectives. Agent environments are classified based on different properties that can affect the complexity of the agent's decision-making process (Russell and Norvig, 1995). They include:

- *Accessible vs. inaccessible*
  - An accessible environment is one in which the agent can obtain complete, timely and accurate information about the state of the environment. The more accessible an environment, the less complicated it is to build agents to operate within it. Most moderately complex environments are inaccessible.
- *Deterministic vs. non-deterministic*
  - Most reasonably, complex systems are non-deterministic – the state that will result from an action is not guaranteed even when the system is in a similar state before the action is applied. This uncertainty presents a greater challenge to the agent designer.
- *Episodic vs. non-episodic*
  - In an episodic environment, the actions of an agent depend on a number of discrete episodes with no link between the performance of the agent in different scenarios. This environment is simpler to design since there is no need to reason about interactions between this and future episodes; only the current environment needs to be considered.
- *Static vs. dynamic*
  - Static environments remain unchanged except for the results produced by the actions of the agent. A dynamic environment has other processes operating on it thereby changing the environment outside the control of the agent. A dynamic environment obviously requires a more complex agent design.
- *Discrete vs. continuous*
  - If there are a fixed and finite number of actions and percepts, then the environment is discrete. A chess game is a discrete environment while driving a taxi is an example of a continuous one.

Agent examples

A simple example of an agent in a physical environment is a thermostat for a heater. The thermostat receives input from a sensor, which is embedded in the environment, to detect the temperature. Two states: (1) temperature too cold and (2) temperature OK are possible. Each state has an associated action: (1) too cold ➡ turn the heating on and (2) temperature OK ➡ turn the heating off. The first action has the effect of raising the room temperature, but this is not guaranteed. If cold air continuously comes into the room, the added heat may not have the desired effect of raising the room temperature.

Background software processes which monitor a software environment and perform actions to modify it can be viewed as agents. A software daemon that continually monitors a user's incoming e-mail and indicates via a GUI icon whether there are unread messages can also be viewed as a simple agentl.

Agents and Objects

Doesn't object-oriented programming provide these agent features? What does an agent offer that Java can't? After all, objects encapsulate data that can represent the state of the object, have methods that enable the objects to perform actions and can communicate by message passing. Despite these similarities, there are significant differences between agents and objects. An object may be said to exhibit autonomy over its state (by defining its instance variables as private) but it does not exhibit control over its behavior. The designers of an object oriented system work towards a common goal – if an object O_i invokes method m of object O_j then that method will be executed as the designers have ensured that it is in the best interest of the system. In many types of multi-agent systems, where agents may be built by and/or for different and competing organizations, no such common goal can be assumed. Thus, the agent decides whether to execute the requested method based on its design goals. "Objects invoke, agents request" or as Wooldridge (1995) indicates he has heard it said "Objects do it for free; agents do it for money".

A second important distinction is that objects do not inherently have anything to say about how to build a system that integrates flexible autonomous behavior. Of course, there is no reason that such systems could not be built with objects but the standard object-oriented programming model has nothing to do with these types of behavior.

Finally, a multi-agent system is intrinsically multi-threaded (each agent is assumed to have at least one thread of control). Object-oriented languages may enable multi-threading but autonomy is not a sine qua non.

Agents and Expert systems

What about expert systems? Couldn't they be considered as agents? Expert systems typically do not exist in an environment – they are disembodied. They obtain their information not through sensors but through a user acting as a middle man. MYCIN, the expert system whose purpose was to assist physicians in the treatment of blood infections in humans, acted as a consultant – it did not operate directly on humans or any other environment. Similarly, expert systems do not act on any environment but instead give feedback or advice to a third party. In addition, expert systems are generally not required to be capable of cooperating with other expert systems. This does not mean that an expert system cannot be an agent. In fact, some real-time (typically process control) expert systems are agents.

INTELLIGENT AGENTS

The idea of intelligent software agents has captured the popular imagination. Tell the agent what you want done, set it free, and wait for it to return results sounds too good to be true. But we'll come back to that later. In the meantime, let's address the question of what makes an agent intelligent. Wooldridge and Jennings (1995) define an intelligent agent as one that is capable of flexible autonomous action to meet its design objectives. Flexible means:

- *reactivity*: intelligent agents perceive and respond in a timely fashion to changes that occur in their environment in order to satisfy their design objectives. The agent's goals and/or assumptions that form the basis for a procedure that is currently executing may be affected by a changed environment and a different set of actions may be need to be performed.

- *pro-activeness*: reacting to an environment by mapping a stimulus into a set of responses is not enough. As we want intelligent agents to do things for us, goal directed behavior is needed. In a changed environment, intelligent agents have to recognize opportunities and take the initiative if they are to produce meaningful results. The challenge to the agent designer is to integrate effectively goal-directed and reactive behavior.

- *social ability*: intelligent agents are capable of interacting with other agents (and possibly humans), through

negotiation and/or cooperation, to satisfy their design objectives.

Other properties sometimes mentioned in the context of intelligent agents include:

- *mobility:* the ability to move around an electronic environment
- *veracity:* an agent will not knowingly communicate false information
- *benevolence:* agents do not have conflicting goals and every agent will therefore always try to do what is asked of it
- *rationality:* an agent will act in order to achieve its goals insofar as its beliefs permit
- *learning/adaptation:* agents improve performance over time

What is driving the interest and need for intelligent agents? Users of the Web are faced with information overload; the amount of data available doubles annually. Individuals can analyze only about 5% of the data and most efforts do not provide real meaning. Thus, the need for intelligent agents is critical to assist in searching, filtering, and deciding what is relevant to the user. Forrester Research (Coolidge, 2001) estimated that by the year 2005, 20 million households will be using the Web for investment and financial planning advice – quite an important task for a critical life decision without some means of assistance.

To put these concepts into a reality based framework, here's a scenario of what an intelligent agent might be able to do in the future (Wooldridge and Jennings, 1995). You are editing a file when your PDA requests your attention: an e-mail message has arrived that contains notification about a paper you sent to an important conference and the PDA correctly predicted that you would want to see it as soon as possible. The paper has been accepted and, without prompting, the PDA begins to look into travel arrangements by consulting a number of databases and other networked information sources. A short time later, a summary of the cheapest and most convenient travel options is presented to you for selection and approval.

Know how, with no how

One way to convey to an agent the task it should perform is to simply write a program that the agent should execute. The agent will do exactly as told and no more - if an unforeseen circumstance arises the agent will have no clue as to how it should react. Thus, what we really want is to tell our agent *what* to do without really telling it *how* to do it. Associating a performance measure or utility with each state is one such technique. A utility is a number representing the 'goodness' of the state – the higher the utility the better the state. The task of the agent is to maximize utility without being told how to do so.

An example of the use of such a utility function is in Tileworld (Pollack, 1990). Tileworld is a simulated, two-dimensional grid environment, both dynamic and unpredictable, which contains agents, tiles, obstacles and holes. An agent can move in four directions – up, down, left or right – and if it is located next to a tile, it can push it. An obstacle is a group of immovable grid cells; agents are not allowed to travel freely through obstacles. An agent scores points by filling holes with tiles, the aim being to fill as many holes as possible. A number of parameters can be set, including the rate of appearance and disappearance of holes, obstacles, and tiles. The performance of an agent on a run r is defined as the number of holes filled in run r divided by the number of holes that appeared in run r. Despite its seeming simplicity, Tileworld enables the study of a number of important capabilities of agents. Chief among them is the ability of an agent to react to changes in the environment and to exploit opportunities when they arise. If an agent is pushing a tile to fill a hole and the hole disappears before being filled, the agent should realize its original goal is no longer in effect and 'rethink' its objective by searching for a new hole to fill. In a similar vein, if an agent is pushing a tile to fill a hole that's four grid cells in the north direction and an empty hole suddenly appears one cell to the east of the agent's current position, the agent should capitalize on this change and fill the closer hole. All other things being equal, the chances of the hole to the east not disappearing in one move is four times greater than the hole to the north which is four moves away. Tileworld represents an oversimplification of real-world scenarios but it is a useful environment for experimentation.

But how do they do it?

How is this know-how incorporated into software? Shoham introduced a new programming paradigm based on societal views of computation that he called agent-oriented programming (Shoham, 1993). He called the programming language AGENT0. The key idea is programming agents in terms of mentalistic notions such as belief, desire and intention (BDI), which have been developed by agent theorists to represent the properties of agents. In AGENT0, an agent is specified in terms of a set of capabilities (things the agent can do), a set of initial beliefs, a set of initial commitments (an agreement to perform a particular action at a particular time) and a set of commitment rules. Capabilities are used by the agent to decide whether to

adopt commitments; an agent will not adopt a commitment to perform an action if the agent can never be capable of performing that action.

The set of commitment rules determines how the agent acts. Each commitment rule contains a message condition, a mental condition and an action. In order to determine whether such a rule fires, the message condition is matched against the message the agent has received and the mental condition is matched against the beliefs of the agent. If the rule fires, the agent becomes committed to performing the action. For example, agent A sends a commitment request in a message to agent B. Agent B will accept or reject the request based on the details of the request, its behavioral rules, and current mental model. B will then send a message to A indicating acceptance or rejection of the request. If B accepts the request, it agrees to attempt to perform the requested action at the requested time if possible. For example, agent B may have committed itself to make an inquiry into a database on behalf of A. Even if B has the ability to connect and query the database, it may not be possible at the specified time due to a disk crash during the database access. B will monitor the execution of the query and send a message back to A to report success or failure of the commitment.

THE NEXT STEP: MULTI-AGENT SYSTEMS

As the field of AI matured, it broadened its goals to the development and implementation of multi-agent systems (MASs) as it endeavored to attack more complex, realistic and large-scale problems which are beyond the capabilities of an individual agent. (Sycara, 1998). The capacity of an intelligent agent is limited by its knowledge, its computing resources, and its perspective (Simon, 1957). By forming communities of agents or agencies (Figure 2), a solution based on a modular design can be implemented where each member of the agency specializes in solving a particular aspect of the problem. Thus, the agents must be able to interoperate and coordinate with each other in peer-to-peer interactions. The characteristics of MASs are defined as follows (Sycara, 1998):

- Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint
- There is no global control system
- Data are decentralized
- Computation is asynchronous

What can MASs do that has generated such interest in them? First, they can be used to solve problems that are too large for a centralized agent to solve because of resource limitations and/or to avoid a one point bottleneck or failure point. Secondly, to keep pace with changing business needs legacy systems, which may not be able to be rewritten due to a combination of cost, time, and technical know-how, can be made to interoperate with other agents in an agent society by building an agent wrapper around them (Genesereth and Ketchpel, 1994). In addition, those agencies which are not self-contained but interact with outside agents e.g., buying and selling, contract negotiation, meeting scheduling (Garrido and Sycara 1996), are by nature MASs. Finally MASs enhance performance in the following areas: (1) computational efficiency through concurrency, (2) reliability via redundancy, (3) extensibility of the agency by changing the number and capabilities of the agents, (4) maintainability via modularity and (5) reuse of agents in different agencies to solve different problems.

Figure 2. Multi-Agent System

MASs sound great, but has anyone implemented one that is useful? Here is but a small sample of applications:

- ARCHON (ARchitecture for Cooperative Heterogeneous ON-line systems) (Jennings et al, 1996a; Parunak, 1999) is one of the largest and probably best known European multi-agent system development project to date. Multi-agent technology was developed and deployed in a number of industrial domains – the most significant being a power distribution system currently operational in northern Spain for the electricity utility Iberdrola. The ARCHON technology was subsequently deployed in particle accelerator control for CERN (Perriolat et al, 1996).
- In workflow and business process control, the ADEPT system (Jennings et al, 1996b) models numerous departments at British Telecom involved in installing a network to deliver a particular type of telecommunications service. Beginning with the initial customer contact, followed by customer vetting, requirements definition, determination of legality of service, design plan, and final quote, departments and individuals within the departments are modeled as agents that negotiate with each other to reach a mutually agreeable contract for the customer.
- The OASIS system (Optimal Aircraft Sequencing using Intelligent Scheduling) (Ljungberg, 1992) is an air-traffic control system whose purpose is to assist an air-traffic controller in managing the flow of aircraft at an airport. OASIS contains (1) global agents which perform generic domain functions e.g., arranging the landing sequence of aircraft and (2) an aircraft agent for each aircraft in the system airspace. It was trialed at Sydney airport in Australia.
- A proliferation in online auctions led to the need to monitor and bid in multiple auctions to procure the best deal for the desired item. Both of these actions are complex and time consuming, particularly when the bidding times for different auctions may or may not overlap and when the bidding protocol may differ. Anthony and Jennings (2003) describe the development of a heuristic decision making framework that an autonomous agent can exploit in such situations. An agent-based architecture for bidding on the New York Stock Exchange has also been proposed (Griggs, 2000) as well as trading simulation that merges automated clients with real-time, real-world stock market data (Kearns and Ortiz, 2003).

THE DOWNSIDE

Despite the significant advances made in the science of agent systems, the pragmatic engineering of such systems is not as well understood (Wooldridge and Jennings, 1998). Some of the common pitfalls include:

- Agents do not make the impossible possible – they are not a magical problem solving paradigm
- Agents are not a universal solution; there are situations where a conventional software development paradigm (e.g., object oriented) may be far more appropriate
- Projects that employ agents because of the hype about agents but with no clear picture of what benefits agents will bring to the project are likely doomed to failure.
- Building a successful prototype with agents does not guarantee it will prove scalable and reliable in solving the full blown, real-world problem.
- The design does not leverage concurrency
- There are too few or too many agents
- A multi agent system can not be developed successfully by throwing together a number of agents and letting the system run – anarchy may result. Instead, a great deal of a priori system-level engineering is required especially for large-scale systems.

Even with a well-designed and implemented MAS, other issues can prevent the acceptance of the system by the user community:

- Cost justification: is it worth the price?
- Security: will data be secure, particularly in a distributed environment? Will an agent respect restrictions from other servers and go only where allowed?

- Legal/Ethical issues: will agents' goals and plans be designed so as not to do anything illegal or unethical? Are there guidelines as to what determines a well- behaved agent? With whom does liability rest for the decisions, actions and/or recommendations of these systems? (Mykytyn et al, 1990).
- Accuracy: can the correctness of the results be guaranteed?
- Acceptance by society: An impediment to the widespread adoption of agent technology is social – for individuals to be comfortable with delegating tasks they must first have trust in agents. (Bradshaw, 1997)

CONCLUSION

Agent-based systems technology is a vibrant and rapidly expanding field of academic research and business world applications. By providing a new paradigm for designing and implementing systems for a complex, dynamic and distributed environment where the common currency is negotiation, unplanned for events can be managed in a way that is beneficial to the overall system performance. Agent technology is greatly hyped as a panacea for the current ills of system design and development, but the developer is cautioned to be aware of the pitfalls inherent in any new and untested technology. The potential is there but the full benefit is yet to be realized. Agent technology will achieve its true potential only if users understand its business value (Radjou, 2003). Much work is yet to be done.

REFERENCES

1. Anthony, P. and Jennings, N.R. (2003) Developing a Bidding Agent for Multiple Heterogeneous Auctions, *ACM Transactions on Internet Technology*, (3)3, pp. 185-217.
2. Bradshaw, J. (1997) Software Agents, Menlo Park, CA, AAAI Press.
3. Brustolini, J.C. (1991) Autonomous Agents: Characterization and Requirements, Report CMU-CS-91-204, School of Computer Science, Carnegie-Mellon University, Pittsburgh, 1991.
4. Coen, M.H., SodaBot: A Software Agent Construction System, MIT AI Lab, USA, 1995.
5. Coolidge, C. (2001) Financial Planning, *Forbes – Best of the Web Spring 2001*, 167(5), p. 84.
6. Etzioni, O. (1996) Moving up the information food chain: deploying softbots on the World Wide Web*, Proceedings of the 13th national Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, pp.4-8.
7. Franklin S.P. and Graesser A.G. (1996) Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents, In: *Intelligent Agents III*, J.P. Muller, M.J. Wooldridge & N.R. Jennings (Eds.), Springer-Verlag, Heidelberg, 1996.
8. Gilbert, D., Aparicio, M., Atkinson, B., Brady, S., Ciccarino, J., Grosof, B., O'Connor, P., Osisek, D., Pritko, S., Spagna, R. and Wilson, L. (1995) IBM Intelligent Agent Strategy, White Paper, 1995.
9. Griggs, K. (2000) An Agent oriented Business Model for E-Commerce based on the NYSE Specialist System, *ACM SIGCPR 2000*, Evanston, Illinois, pp.136-143.
10. Hayes-Roth, B., Pfleger, K., Lalanda, P., Morignot, P. and Balabanovic, M. (1995) A Domain Specific Software Architecture for Adaptive Intelligent Agents, *IEEE Transactions on Software Engineering*, 21(4), PP.288-301, April 1995.
11. Honavar, V. (1999), Intelligent Agents and Multi Agent Systems, IEEE CEC.
12. Jennings, N. R., Mamdani, E. H., Corera, J., Laresgoiti, I., Perriolat F., Skarek, P. and Varga, L. Z. (1996a) Using ARCHON to develop real-word DAI applications, *IEEE Expert*, 1996, 11 (6), pp. 64-70.
13. Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., O'Brien, P., Wiegand, M. E (1996b) Agent-based business process management, *International Journal of Cooperative Information Systems*, 5 (2&3), 1996, pp. 105-130.
14. Kearns, M. and Ortiz, L (2003) The Penn-Lehman Automated Trading Project, *IEEE Intelligent Systems*, Nov/Dec 2003, pp. 22-31.
15. Kendall, E.A., Murali Krishna, P.V., Suresh, C.B. and Pathak, C.V. (2000), An application Framework for Intelligent and Mobile Agents, *ACM Computing Surveys*, Vol. 32, No. 1es, March 2000.
16. Ljungberg, M., Lucas, A. (1992) The OASIS Air Traffic Management System, *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, PRICAI '92, Seoul, Korea.

17. Maes, P. (1995) Artificial Life Meets Entertainment: Lifelike Autonomous Agents, Communications of the CAN, Special Issue on Novel Applications of AI.
18. Mykytyn, K., Mykytyn, P., Slinkman, C. (1990) Expert Systems: A Question of Liability?, *MIS Quarterly*, March 1990, pp. 27-42.
19. Parunak, H.V.D. (1999) Industrial and Practical Applications of DAI, Multi-Agent Systems, pp. 377-421, MIT Press, Cambridge, MA.
20. Perriolat, F., Skarek, P., Varga L. Z, and. Jennings, N. R (1996) *Using Archon, Part 3: Particle Accelerator Control*, IEEE Expert, 1996, 11 (6), pp. 80-86.
21. Pollack, M.E. and Ringuette, M. (1990) Introducing the Tileworld: Experimentally Evaluating Agent Architecture, *Proceedings of the 8th National Conference on Artificial Intelligence* (AAAI-90), Boston, MA, pp. 183-189.
22. Radjou, N. (2003) Software Agents in Business: Still An Experiment, *Forrester Research Brief*, March 27. 2003.
23. Russell, S. and Norvig, P. (1995) Artificial Intelligence: A Modern Approach, Prentice-Hall, Englewood Cliffs, NJ.
24. Shoham, Y. (1993) Agent Oriented Programming, *Journal of Artificial Intelligence,* 60 (1), pp. 51-92, 1993.
25. Simon, H., (1957) Models of Man: Social and Rational –Mathematical Essays on Rational Human Behavior in a Social Setting, Wiley, New York.
26. Sycara, K.P. (1998) Multiagent Systems, *Artificial Intelligence*, Summer 1998, pp. 79-92
27. Turban, E. and Aronson, J.E. (2001) Decision Support Systems and Intelligent Systems 6/e, Prentice Hall, Upper Saddle River, NJ.
28. Wooldridge, M. (2002) An Introduction to MultiAgent Systems, Wiley, England,
29. Wooldridge, M. and Jennings, N.R. (1995) Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review,* 10(2), pp.115-152.
30. Wooldridge, M. and Jennings, N.R. (1998) Pitfalls of Agent-Oriented Development, *Proceedings of the 2nd International Conference on Autonomous Agents* (Agents 98), Minneapolis/St. Paul, MN, pp, 385-391.