Multiagent System for Monitoring Chronic Diseases

D. Floroian[1], L. Floroian[2], and F. Moldoveanu[1]

[1] Transilvania University of Brasov/Automatics Department, Brasov, Romania
[2] Transilvania University of Brasov/Physics Department, Brasov, Romania

Abstract— **The aim of this paper is to present a society of intelligent agents linked into a multiagent system with the scope of monitoring a large group of human beings in order to detect possibilities of estimating catching chronic diseases. For this purpose a heterogeneous group of patients was taken into observation into approximately 15 years period. The group is provided by a family doctor and is formed by many social categories, many age increments and some different cities. The paper will prove that such monitoring will detect in the first stage or will detect the possibilities of being ill based only on historical dates of the patients. Obviously the names of the patients were not taken into multiagent system, but in real cases the family doctor could prevent a real patient about the danger of his life style or medical historical dates.**

Keywords— **intelligent agents, chronic diseases, multiagent system, health care.**

I. Introduction

Nowadays the chronic diseases are increasing and the associated problems (cost of medications, cost of medical examinations, social exclusions, etc.) are more and more expensive. In the same time the number of persons with chronic diseases increases year by year. It is obviously interesting to find a way to prevent such things to happen in the case that we could determine what type of behavior or what type of previous disease leads to a chronic disease.

Furthermore, if will be possible to have an automatic equipment (or software program in this case) to detect the formation of chronic diseases, will be possible (at least theoretical) to reduce the number of patients with chronic diseases.

At Brasov County official dates reveal the fact that about 20% of population is in a prediabetes stage, according to national statistics. Another 10% of population has an immediate risk of heart attack. This situation implies a sustained effort to prevent and detect the chronic diseases.

This paper presents a study made on 1874 patients from different locations, which evolve on a 15 year period. In this purpose each patient was agentified using a program computer. For each person in the group there is an intelligent agent who takes the basic information about patient and in the same time, the medical information about that patient. In the real life these information are connected with the name of the real patient. In our study the name was replaced by numbers for medical confidence because the medical dates are real.

Furthermore each agent enrolls it to a multiagent system. The multiagent systems provide possibility of the agents' collaborations and the interaction with the user interface. Each agent will provide needed information on demand.

Our program will display based on agents the evolution of different diseases along these years for this group of patients. In the future we want to develop the program in order to automatically alert the family doctor about the chronic diseases which will potentially develop for a specific patient. This is possible because the multiagent system learns some basic behaviors about information that it have.

The paper is organized as follow: in second section we present the supporting concepts as "multiagent system" and "intelligent agents"; in section III we present an example with 1874 group of patients and connection between chronic diseases and them. Finally in conclusion section we present the results.

II. Supporting concepts

A. *Multiagent System*

In the context of this paper, an agent is a computer program associated either to some hardware component (a sensor connected to a person) or to some specific information about a person and which can run continuously and autonomously. The multiagent system (MAS) provides a set of individual agents who can communicate between them for achieving a purpose and with a central station for monitoring process. The agents within the system model interact with the agents who represent the components or operations to find the needed information with minimal resources and in optimal time.

The agents' interaction is done under the control of the coordination agent (which also provides the graphic user interface) and looking for key information. When a satisfactory solution cannot be found in the current structure it will try to looking for the alternative options with gradually relaxed constraints, allowing subsystems to regroup. This

S. Vlad and R.V. Ciupa (Eds.): MEDITECH 2011, IFMBE Proceedings 36, pp. 26–31, 2011.
www.springerlink.com

reflects the Belief Desire Intention – BDI architecture (together with the feedback for system states updates). The basic requirements for the MAS architecture are that the system model must supply enough information about the patient to allow the planning, scheduling, and reconfiguration of the dates to be carried out under gradually relaxed constraints. The model also needs to avoid centralized control which does not characterize the multiagent technology, in order to allow subsystems to autonomously negotiate each other and to collaborate across system boundaries to form new groupings [1].

Also the subsystems must be able to carry out collectively simulation of discrete events, in any grouping, in order to evaluate restructured decisions. Based on the analysis, each subsystem on the system hierarchy needs to be modeled as an agent. An agent should be able to send and receive messages to or from other agents and the environment, and to make decisions to take part in bidding and negotiation. There should also be a mechanism to support dynamic simulation of discrete events [2, 3].

In order to implement the described MAS, the BDI architecture is very convenient because the environment is not fully known and it is changes over time. Epistemic Deontological Axiological (EDA) architecture is not so useful because the norms which are included will not help in this situation. Also Layered architectures or Logic Based Agent architectures are too close to a centralized control in described situations [4, 5].

In this paper we consider that the communications between agents are FIPA compliant (Foundation for Intelligent Physical Agents) like described in [6]. Also for MAS implementation we use JADE (Java Agent DEvelopment Framework) which is a free tool available online [7]. Furthermore the ontology was integrated in JADE with Protégé 2000 [8, 9, 10, 11].

For success implementing the BDI architecture a belief revised (BR) function is necessary. This function is intended to revise the knowledge database (the beliefs of the agents) in order to maintain the consistency of the database. In our special case this function is necessary because the beliefs of the agents could be wrong in case of an unpredictable situation is involved in FMS flow. In these cases the BR function will correct the beliefs of the agents. The feedback flow of the BR function is represented in figure 1 (adapted from [12]).

As is seen in the figure 1, the agents of the MAS can perceive only a part of the variables of the patients. Using the beliefs gathered initially from the patient's model and the new perceived information, the BR function rebuilds the unperceived information. Together with the update rebuilt information these knowledge represent the new set of beliefs of the agents.

For this approach we focus for a MAS where the environment is a dynamic system with a fixed structure where the value of some system variables (necessary for BR function implementation) is unknown. In this paper we consider a practical case study for implementing such concepts.

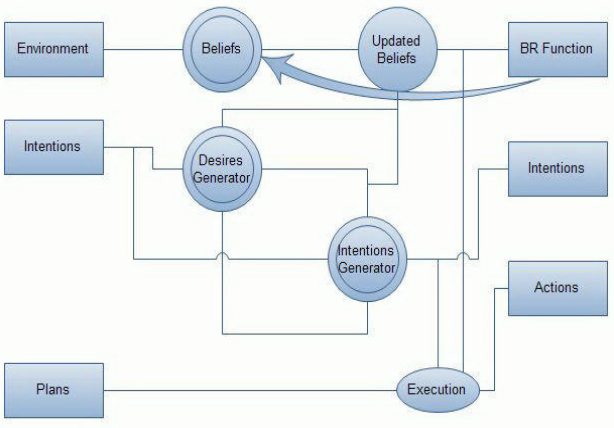

Fig. 1 The BR function generates an update belief

B. *Patients Agentifying*

For integrating of the information in MAS we need a procedure for connecting of the information with MAS's coordination agents. This procedure is called patient agentifying and implies that for each patient in the group an agent to be created. That agent must retain both medical and personal information. The personal information is needed for infrastructure purpose and also for statistical dates. The medical information is needed for experiment. As we mentioned before the name are replaced by number in this study case for confidential purpose. In the real cases the names must be accessible only to the medical or authorized staff.

This procedure is difficult and takes a long time. But this procedure is needed only one time. After this all will happen automatically. The user must complete or update the information only when it is necessary [13].

After agentifying process the MAS could access all information that it needs. The access to the information is made only by agent's way. In this stage we have a collection of intelligent agents that cooperate in order to achieve their purpose [4].

In this process the user has a graphic user interface (GUI) that helps the input of the information. This GUI is represented in figure 2. The GUI will provide information that is needed for the agentifying process. The action for the submit button implies that an agentifying agent is solicited to complete the agentifying process. By this action all the information is used to setup the patient's agent. Thus a specific agent associated to a specific patient is obtained. This process is called agentifying of patients [14, 15, 16, and 17].

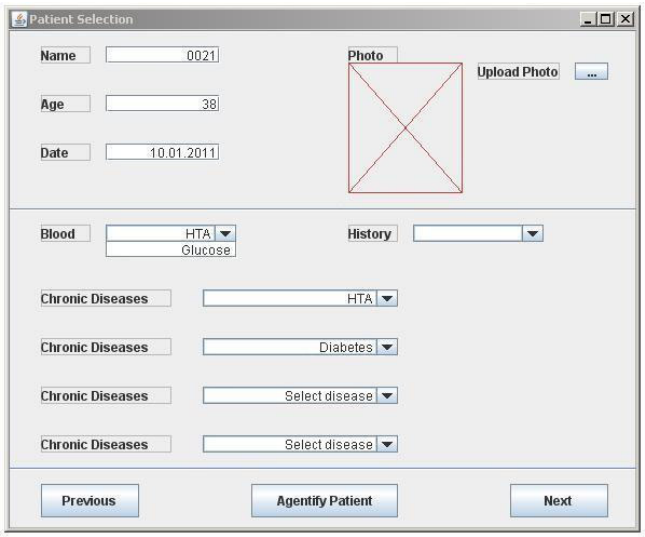

Fig. 2 The GUI interface

GUI is configurable in order to permit many kinds of diseases to be monitored.

In our MAS the cooperation and coordination between PTAxs agents is important because they share the same goals. In the same time is important to notice that the mechanism that governs coordination is the interaction protocols.

Cooperation (to achieve a certain goal) is based on two phases. First decompose the main task into several small tasks (i.e. finding diabetes mellitus is decomposing in: search for already declared patients, search for inducing diabetes diseases, search for other emerging factors, reduce the redundancies, compose the outputs). Several decomposition plans can be generated according to the available resources and the capabilities of the cooperating entities. In the second phase each subtask is executed and after all the *generateOutput* task will contain the results. These entire subtasks are running concurrently. This implies a competition between agents because each of them needs to obtain their task and also report information to others. The competition is regulated by a coordination agent in order to keep the accuracy of dates.

III. Study case

In order to prove the above concepts we use a specific group of 1874 patients. These patients are from different four locations (near Brasov).

They are monitored over a 15 years period of time for chronic diseases. We want to obtain some results which will indicate the chronic diseases rates and to obtain a link that we could use in the future to predict the occurrence of chronic diseases in a specific population.

In our study case we choose to monitor: diabetes mellitus, dyslipidemia, hypertension, obesity, ischemic cardiomyopathy (see fig. 3).

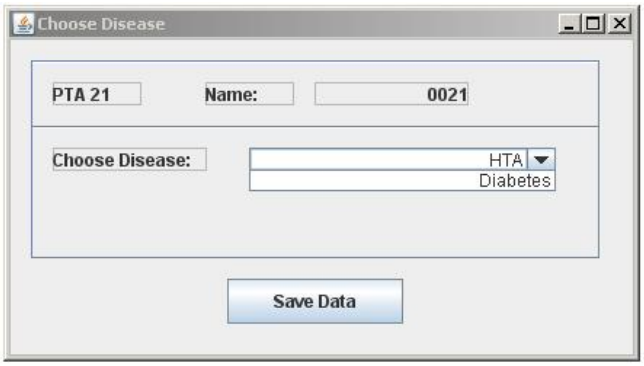

Fig. 3 Chosen the chronic disease

In our case a patient's agent will retain only information about age, chronic diseases and historical dates. However it could be programmed to retain much more information.

This study case will only take care about a few chronic diseases because its role is to prove the procedure. In the future we want to extend this example to permit prediction of all chronic diseases occurrence.

In the first stage we agentified all 1874 patients, using the GUI presented in figure 2. After this process we obtained 1874 agents. These agents are coordinated by a coordination agent (CA). This agent has an interface role between patient's agents and MAS. Also this agent preserves the accuracy of dates by rules (these rules modify the behavior of the agents). There is not a subordinating relationship but a cooperative relation. In this way the user could obtain the results, the agents could communicate and the MAS principle is not broken [1].

The MAS is structured like in figure 4. There are 1874 patient agents (PTAx) and 1 CA. These agents communicate each other's in order to cooperate. Also PTAxs communicate with CA. For interaction with people there is a GUI (see fig. 2).

For the study of the MAS's agents we use JADE's GUI. This interface reveals the agents that constitute the MAS. Also the JADE has a core that runs silent in background. Only at specific request the GUI is displayed (see fig. 5).

For our purposes, we have adopted the description of an agent as a software program with the capabilities of searching and cooperate. This implementation is made in JADE because this development tools are very versatile and could be very well integrated with others development tools

(like Protégé-2000 and Java). Also JADE is an open source FIPA compliant Java based software framework for the implementation of multiagent systems. It simplifies the implementation of agent communities by offering runtime and agent programming libraries, as well as tools to manage platform execution and monitoring and debugging activities. These supporting tools are themselves FIPA agents. JADE offers simultaneously middleware for FIPA compliant multiagent systems, supporting application agents whenever they need to exploit some feature covered by the FIPA standard (message passing, agent life cycle, etc.), and a Java framework for developing FIPA compliant agent applications, making FIPA standard assets available to the programmer through Java object-oriented abstractions.

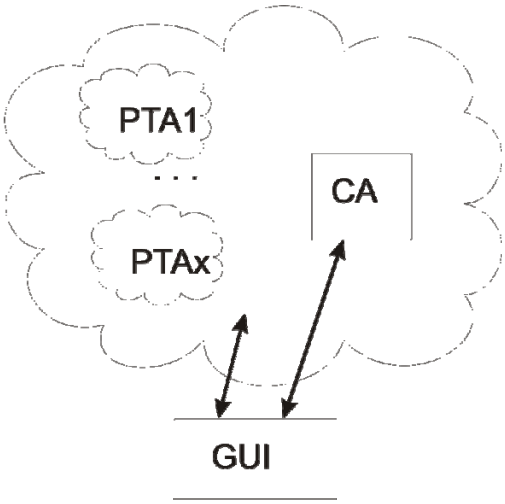

Fig. 4 MAS architecture

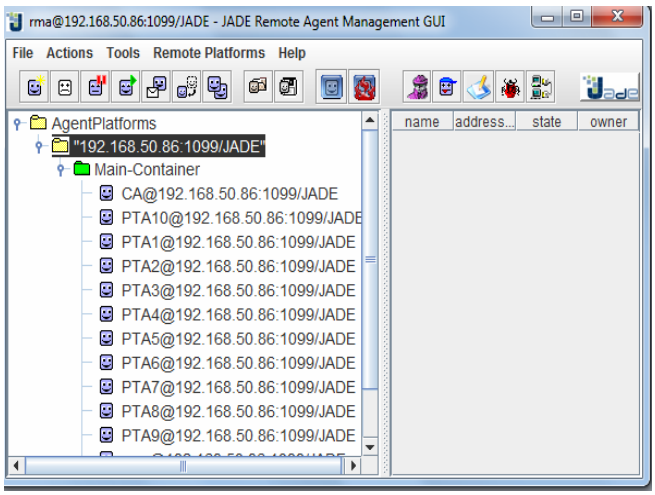

Fig. 5 MAS as seen in JADE

The general management console for a JADE agent platform (RMA – Remote Monitoring Agent), like in figure 5, acquires the information about the platform and executes the GUI commands to modify the status of the platform (creating new agents, shutting down containers, etc.) through the AMS (Agent Management System).

The agent platform can be split between several hosts (provided that there is not firewall between them). The agents are implemented as one Java thread and Java events are used for effective and lightweight communication between agents on the same host.

The parallel tasks can be still executed by one agent, and JADE schedules these tasks in a more efficient (and even simpler for the skilled programmer) way than the Java Virtual Machine (VM) does for threads.

Several Java VMs, called containers in JADE, can coexist in the same agent platform even though they are not running in the same host as the RMA agent. This means that a RMA can be used to manage a set of VMs distributed across various hosts.

Each container provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host.

The DF (Directory Facilitator), AMS, and RMA agents coexist under the same container (main-container) together with the patient's agents. To facilitate message reply, which, according to FIPA, must be formed taking into account a set of well-formed rules such as setting the appropriate value for the attributes in-reply-to, using the same conversation-id, etc., the method createReply() is defined in the class that defines the ACL (Agent Communication Language) message.

Different types of primitives are also included to facilitate the implementation of content languages other than SL (Standard Languages), which is the default content language defined by FIPA for ACL messages. This facility is made with Protégé 2000 as depicted in figure 6.

Protégé presents also a graphical view of ontology classes which facilitate understanding the fact that from the point of view of the programmer, a JADE agent is simply a Java class that extends the base agent class. It allows inheriting a basic hidden behaviour (such as registration, configuration, remote management, etc.), and a basic set of methods that can be called to implement the application tasks of the agent (i.e. send/receive ACL messages).

Moreover, user agents inherit from their Agent superclass some methods to manage agent behaviours. Also this diagram can be represented in UML (Unified Modelling Language) because behaviours are implemented as hierarchy of classes. The Protégé 2000 connects to PTAxs JADE Agents by including a Protégé configuration file in the Java compiler. The ontology is created in Protégé 2000 and MAS's classes are presented in figure 6.

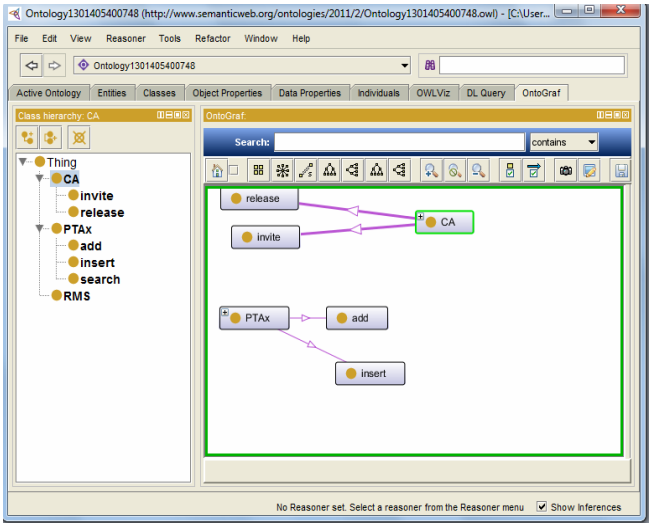

Fig. 6 MAS's ontology realized in Protégé 2000

The outputs are revealed in figures 7, 8, 9, 10 and 11. In these figures are presented the evolutions for diabetes mellitus, dyslipidemia, hypertension, obesity, ischemic cardiomyopathy for PTAxs included in MAS. These figures don't represent simple histograms of the respective illness in time; they are the outputs of the specific PTAx. These outputs reflect the searches and also the incidence based on the other illness declared in patient's clinical dates. The time for searching, cross-over searching and data-link is minimized by the MAS algorithm and is made in one step. This is the advantage of using this MAS. This is possible because the PTAxs are not simple searching tools.

In these figures could be seen that a spectacular growth has occurred in recent years. This problem is a big issue and preventing and detecting the chronic diseases in the first stages are imperative.

In the future we intend to extend the MAS on a very large number of patients for early detecting these diseases based on historical medical dates.

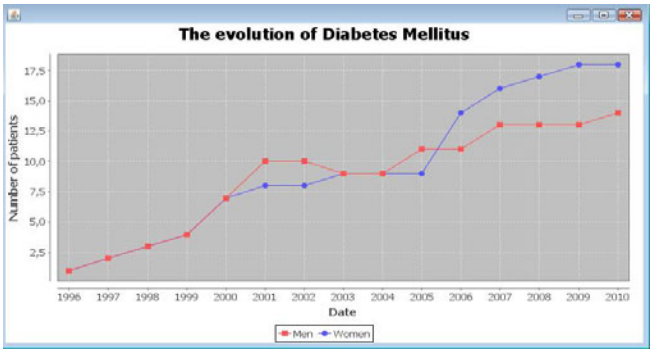

Fig. 7 Diabetic evolution

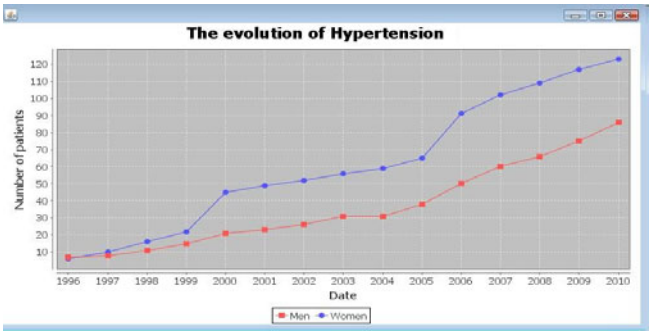

Fig. 8 Arterial hypertension evolution

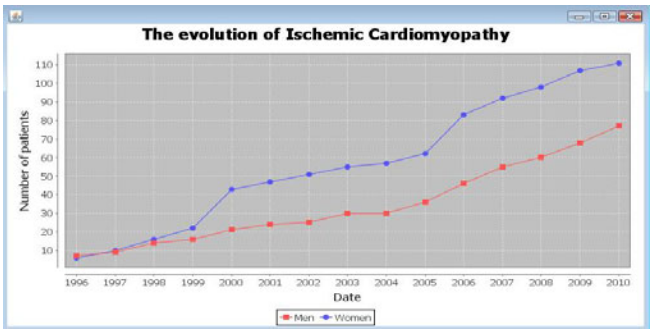

Fig. 9 Ischemic cardiomyopathy evolution

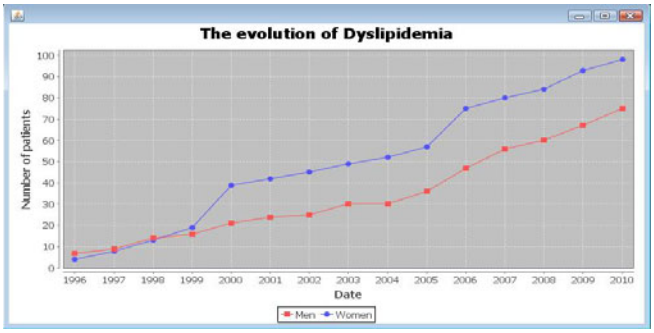

Fig. 10 Dyslipidemia evolution

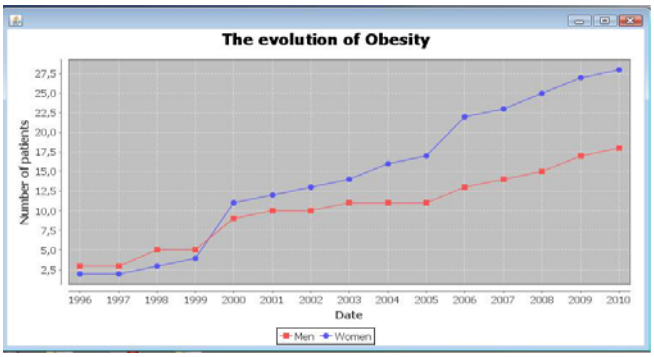

Fig. 11 Obesity evolution

IV. Conclusions

The chronic diseases problem is very serious and the ways to prevent and to detect earlier is also a big issues. This application starts to resolve the second part of the problem. Agent-based computing has great potential in addressing this problem by using distributed resources and reducing the programming effort. In this paper, we present a new approach based on agent-oriented programming and the concept of collaborative intelligent agents.

We develop a multiagent system that consists of patient representative agents. To obtain satisfaction for patients receive, the patient representative agents are designed to be as informative and timely as communicating with people by GUI. In our paper we used Java in the initial pseudo-natural language parser, XML as the knowledge base and XQL as the query language. This paper represents a quite new area and has not been developed to its full capability. Future work includes enhancing the capabilities of agents and the communication, collaboration, and coordination mechanisms in the multi-agent system, improving the pseudo-natural language parser to make it understand the user's input and synonyms better, supporting a more advanced query setup to provide accurate replies in the circumstance of multiple inexact matching, and developing mechanisms for the agents to learn automatically from the users, history profiles, and information on the Web. The ultimate goal is to detect in the first stage the chronic diseases develop.

Also the use of MAS is imperative because of its flexibility. Thus the system can be very easily switched for other diseases and the time for obtaining results remain very short even in the case of millions of patients.

References

1. Floroian D (2009) Multiagent Systems, Albastra, Cluj-Napoca, Romania (in Romanian)
2. Barata J, Camarinha-Matos L, Onori M (2005) A Multiagent Based Control Approach for Evolvable Assembly Systems. Industrial Informatics, INDIN '05. Proc. of the 3rd IEEE Int. Conf., 2005, 478-483
3. Zhang D Z, Anosike A, Lim M K (2007) Dynamically Integrated Manufacturing System (DIMS) – A Multiagent Approach. IEEE Trans. Syst., Man, & Cybern. – Part A, Syst., Humans, 37(5), 824-850
4. Wooldridge M J, Jennings N R. (1995) Agent Theories Architecture and Languages: A Survey, Intelligent Agents. In Proc. Workshop on Agents Theories, Architecture and Languages (ECAI-94), Amsterdam, Holland, 1995, pp. 1–39
5. Jennings N R (2000) On Agent-based Software Engineering. In Artificial Intelligence, no. 117, pp. 277-296
6. FIPA Communicative Act Library Specification (XC00037J). FIPA - Foundation For Intelligent Physical Agents. At http://www.fipa.org/specs/fipa00023/index.html
7. JADE at http://jade.tilab.com/
8. Protégé-2000 at http://protege.stanford.edu
9. Ontoprise: Semantics for the Web at http://www.ontoprise.de
10. The Semantic Web Community Portal at http://www.semanticweb
11. WebOnto. Knowledge Media Institute (KMI), The Open University at http://eldora.open.ac.uk: 3000/webonto/
12. Barata J, Camarinha-Matos L (2003) Coalitions of Manufacturing Components for Shop Floor Agility - the CoBASA Architecture. Int. J. of Networking and Virtual Organisations, 2(1), 50-77
13. Smirnov A, Pashkin M, Chilov N, et al. (2003) Knowledge Source Network Configuration Approach to Knowledge Logistics. Int J General Systems, Vol. 32(3), pp. 251-269
14. Walczak S (2003) A Multiagent Architecture for Developing Medical Information Retrieval Agents. J. Med. Syst., Vol. 27, No. 5, 479-498
15. Shang Y, Shi H (1999) A Web-based multi-agent system for interpreting medical images World Wide Web 2 (1999) pp.209–218
16. Bousquet F, Le Page C. (2004) Multi-agent simulations and ecosystem management: a review. Ecol. Model. 176, 313–332
17. Linard C, Ponc on N, Fontenille D, Lambin EF (2008) A multiagent simulation to assess the risk of malaria reemergence in southern France. Ecolog. Mod. 220:160–174. DOI:10.1016/j.ecolmodel. 2008.09.001

Author: Dan Floroian
Institute: Transilvania University of Brasov
Street: 29, Eroilor Blvd.
City: Brasov
Country: Romania
Email: dan.floroian@unitbv.ro